

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



## THESIS

**PREDISTORTION OF  
QUADRATURE AMPLITUDE MODULATION  
SIGNALS USING  
VOLTERRA SERIES APPROXIMATION**

by

Michael T. Donovan

December, 1996

Thesis Advisor:

Murali Tummala

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 2

19970701 068

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time reviewing instructions, searching existing data sources gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE December 1996		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE PREDISTORTION OF QUADRATURE AMPLITUDE MODULATION SIGNALS USING VOLTERRA SERIES APPROXIMATION			5. FUNDING NUMBERS	
6. AUTHOR(S) Donovan, Michael T.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/ MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Modern digital communication systems are being called upon to move ever increasing amounts of information over decreasingly available bandwidth. This requires that communication systems employ bandwidth-efficient modulation schemes to conserve bandwidth while moving the information at higher data rates. A major stumbling block to using higher order modulation schemes in long haul communication is the distortion caused by high power amplifiers. These high power amplifiers are required to amplify the signal power to a level that will allow distant receivers to correctly demodulate and decode the information. The distortion caused by the high power amplifiers can render a modulation scheme unusable due to the high symbol error rates which result from the extensive skewing of the modulation scheme's signal constellation. This thesis details a predistortion technique using Volterra series approximation techniques to model the inverse of the high power amplifier's distortion characteristics. A 64 Quadrature Amplitude Modulation (64-QAM) system incorporating a predistorter is used to demonstrate the ability to achieve acceptable bit error rates. The implementation of the inverse model and the communication system is performed in MATLAB. The results show the viability of predistortion of digital data to allow the higher order modulation schemes to be incorporated into communication schemes, increasing the overall data rate while conserving bandwidth.				
14. SUBJECT TERMS High Data Rate Communications, QAM, Volterra Series, Predistortion			15. NUMBER OF PAGES 96	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified		18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified		19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified
				20. LIMITATION OF ABSTRACT UL



Approved for public release; distribution is unlimited.

**PREDISTORTION OF  
QUADRATURE AMPLITUDE MODULATION SIGNALS  
USING VOLTERRA SERIES APPROXIMATION**

Michael T. Donovan  
Major, United States Army  
B.S., Purdue University, 1984

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**

from the

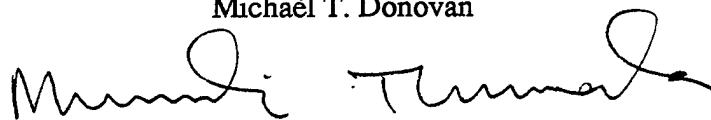
**NAVAL POSTGRADUATE SCHOOL  
December 1996**

Author:

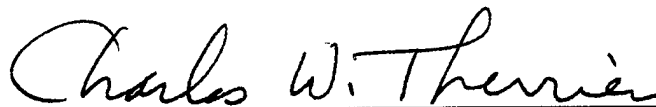


Michael T. Donovan

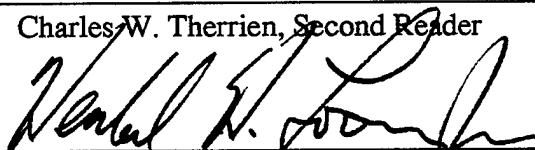
Approved by:



Murali Tummala, Thesis Advisor



Charles W. Therrien, Second Reader



Herschel H. Loomis, Jr., Chairman,  
Department of Electrical and Computer Engineering

DTIC QUALITY INSPECTED 3



## ABSTRACT

Modern digital communication systems are being called upon to move ever increasing amounts of information over decreasingly available bandwidth. This requires that communication systems employ bandwidth-efficient modulation schemes to conserve bandwidth while moving the information at higher data rates. A major stumbling block to using higher order modulation schemes in long haul communication is the distortion caused by high power amplifiers. These high power amplifiers are required to amplify the signal power to a level that will allow distant receivers to correctly demodulate and decode the information. The distortion caused by the high power amplifiers can render a modulation scheme unusable due to the high symbol error rates which result from the extensive skewing of the modulation scheme's signal constellation. This thesis details a predistortion technique using Volterra series approximation techniques to model the inverse of the high power amplifier's distortion characteristics. A 64 Quadrature Amplitude Modulation (64-QAM) system incorporating a predistorter is used to demonstrate the ability to achieve acceptable bit error rates. The implementation of the inverse model and the communication system is performed in MATLAB. The results show the viability of predistortion of digital data to allow the higher order modulation schemes to be incorporated into communication schemes, increasing the overall data rate while conserving bandwidth.



## TABLE OF CONTENTS

I. INTRODUCTION.....	1
II. HIGH POWER AMPLIFIERS .....	3
A. CLASSES OF AMPLIFIERS .....	3
1. Class A .....	3
2. Class B.....	3
3. Class AB .....	4
4. Class C.....	4
5. High Efficiency Amplifiers.....	5
B. INTERMODULATION DISTORTION .....	6
C. INTERMODULATION DISTORTION CORRECTION .....	8
III. DIGITAL MODULATION.....	11
A. QUADRATURE AMPLITUDE MODULATION .....	11
B. QUADRATURE AMPLITUDE MODULATION TRANSMITTER ....	12
C. QUADRATURE AMPLITUDE MODULATION RECEIVER .....	13
D. PROBABILITY OF SYMBOL ERROR .....	14
E. HIGH POWER AMPLIFIER INFLUENCE.....	16
IV. VOLTERRA SERIES MODELING.....	19
A. CONTINUOUS TIME VOLTERRA SERIES .....	19
B. DISCRETE TIME VOLTERRA SERIES .....	20
C. FINITE (pth) ORDER INVERSE .....	23
D. SIMPLIFICATION OF THE VOLTERRA SERIES MODEL.....	24
E. FIXED DATA PREDISTORTER .....	26
F. ADAPTIVE FILTER PREDISTORTER.....	28



1. Recursive Least Squares Estimation .....	29
2. Adaptive Predistorter .....	31
V. SIMULATION RESULTS .....	35
A. FIXED PREDISTORTER .....	35
B. ADAPTIVE PREDISTORTER .....	41
C. COMMUNICATION SYSTEM .....	43
1. Modulator .....	43
2. Demodulator .....	46
3. Signal-to-Noise Ratio .....	48
D. COMMUNICATION SYSTEM PERFORMANCE .....	49
VI. CONCLUSIONS .....	51
A. CONCLUSIONS .....	51
B. FUTURE WORK .....	52
APPENDIX A. MATLAB COMPUTER CODE FOR DETERMINING THE NON- LINEAR DATA COEFFICIENTS .....	53
APPENDIX B. SUPPORTING MATLAB FUNCTIONS CALLED TO DETERMINE THE PREDISTORTER MODEL'S COEFFICIENTS .....	55
APPENDIX C. MATLAB COMPUTER CODE CALCULATING THE FIXED PRE- DISTORTER MODEL .....	57
APPENDIX D. MATLAB COMPUTER CODE FOR DETERIMINING THE PRE- FORMANCE OF A PREDISTORTER MODEL .....	63
APPENDIX E. MATLAB COMPUTER CODE FOR DETERMINING THE ADAP- TIVE PREDISTORTER AND COMMUNICATIONS SYSTEM PER- FORMANCE .....	73
LIST OF REFERENCES .....	85
INITIAL DISTRIBUTION LIST .....	87

# I. INTRODUCTION

This thesis investigates two methods of analyzing the distortion characteristics of High Power Amplifiers (HPA) found in modern digital communication systems. These HPAs can alter the transmitted signal to such an extent it is impossible to decode correctly. The thesis focuses on using digital signal processing techniques to design a predistorter which will result in the emitted signal being nearly intact and capable of proper decoding by the receiver. The MATLAB programming language is used throughout this work to determine the predistorter models and implement them in a 64 Quadrature Amplitude Modulation (64-QAM) digital communication system.

The first predistortion technique investigated is a Volterra series approximation that is based on a Minimum Mean Square Error (MMSE) approach to obtain a fixed predistorter model. The second predistortion technique is a Recursive Least Squares (RLS) adaptive algorithm. The first technique determines an inverse model predistorter which cannot be altered once incorporated into the communication system. The second predistortion technique, however, allows the predistorter to be updated periodically while the communication system is operating.

The HPA serving as the base model for the transmitter is the Traveling Wave Tube (TWT) high power amplifier. The TWT HPA was chosen since it is commonly employed in a large variety of communication systems, and it has a commonly accepted model of its nonlinear distortion characteristics in analytical form. The existence of this analytical model precludes the need to experimentally measure the distortion characteristics of a specific high power amplifier since these characteristics may vary widely within the same family of HPAs.

This thesis focuses on the performance of the predistorter. A highly simplified channel model is used in the simulation of the communication system for this purpose. The MATLAB implementation is designed with the assumption that perfect synchronization of the phase and beginning of each symbol period is achieved. The results of the predistorter

models show how performance varies with the nonlinear order and memory of the Volterra series approximation. The predistorter models are incorporated into a 64-QAM communication system with an additive white gaussian noise (AWGN) channel. The results of the communication system simulations indicate the ability of the predistorter to correct the distortion by quantitatively measuring the probability of symbol error,  $P_s$ , and the effect of various signal-to-noise ratios on the probability of symbol error.

The thesis is organized as follows. Chapter II discusses High Power Amplifiers. Chapter III covers digital communication techniques, specifically the 64-QAM system used in simulations. Chapter IV details the theory of Volterra series, the recursive least squares algorithm, and their application to predistorter modeling. Chapter V outlines the simulation and results of the predistorter approximation. Chapter VI contains the conclusions and areas for further study. The supporting computer code is contained in Appendices A, B, C, D, and E.

## II. HIGH POWER AMPLIFIERS

This chapter discusses the High Power Amplifier (HPA) and its characteristics. The high power amplifier is used to produce signals with sufficient power to be detected and correctly demodulated by the receiver. High power amplifiers are divided into classes based upon the input and output voltage relationship. This influences the power efficiency of the HPAs and results in distortion of the output signal if the amplifier cannot transition during its cycles properly. Specifically, power amplifiers with high power efficiencies can significantly distort a signal during operation. The later sections of this chapter discuss the distortion of these amplifiers.

### A. CLASSES OF AMPLIFIERS.

Amplifiers are employed to scale the voltage and power levels in most electronics systems. These amplifiers are designated by their class of operation. An amplifier's class is judged by its input-output voltage relationship. Some of these classes and an explanation of their input-output voltage relationships are summarized below.

#### 1. Class A.

Class A amplifiers are designed to operate in a linear manner, similar to small signal amplifiers. The distinctive waveform of a Class A amplifier shows the output current flows for the entire cycle (Figure 2.1 (a)). Class A amplifiers are typically used as low level driver amplifiers and in applications where other types of amplifiers cannot be used easily, such as at microwave frequencies. The primary disadvantage of a Class A amplifier is its high quiescent power loss, resulting in a maximum theoretical efficiency of only 50 percent [Refs. 1 and 2].

#### 2. Class B.

Class B amplifiers are biased so that conduction occurs only during one-half, or 180°, of a cycle of the input voltage (Figure 2.1 (b)). These amplifiers can be found in medium and high power linear applications, such as in High Frequency Single Side Band (HF SSB). Class B amplifiers commonly place a pair of active devices (transistors or

vacuum tubes)  $180^\circ$  out of phase with each other such that only one device is active at a time during operation. The efficiency of a Class B amplifier is greater than Class A or AB, a maximum theoretical efficiency of 78.5 percent, but its operation does generate some nonlinear distortion [Refs. 1, 2, and 4].

### 3. Class AB.

Class AB operation is a compromise between Class A and Class B. Its current flows for more than  $180^\circ$  and less than  $360^\circ$  (Figure 2.1 (c)). This compromise yields increased efficiency over the Class A amplifier and less harmonic distortion than the Class B amplifier [Ref. 3].

### 4. Class C.

Class C amplifiers have a higher bias than Class B, well beyond cutoff. This biasing causes their conduction to occur for less than one-half, or less than  $180^\circ$ , of a duty cycle (Figure 2.1 (d)). Typically, the conduction interval is in the range of  $120^\circ$  -  $150^\circ$  of biasing, and the power efficiency is in the range of 60 - 80 percent with the active device in or near the saturation region. This makes them ideal for RF transmission applications.

Class C amplifiers are found primarily at the transmitter end of long haul communication systems where high output power levels are required, but proportionality between the input and output voltages is not critical. For example, they are employed as radio and television transmitters of 50 kW (or more) and LORAN Navigation Stations of 1 MW. Class C high power amplifiers may employ solid state or vacuum tube active devices. Vacuum tubes are the workhorses of high power applications.

Class C amplifiers are able to achieve high efficiency if the duration of the current flow is very small, such as when the voltage drop between the cathode and the plate is at its lowest. Practical applications require a trade off between the amplifier's efficiency and its power output since lower voltage drops reduce the input and output power. Transistors employed as Class C amplifiers have similar operations but are limited in their power and frequency ranges. Transistors are principally found in low power applications since care must be taken not to exceed their current limit [Refs. 4 and 5].

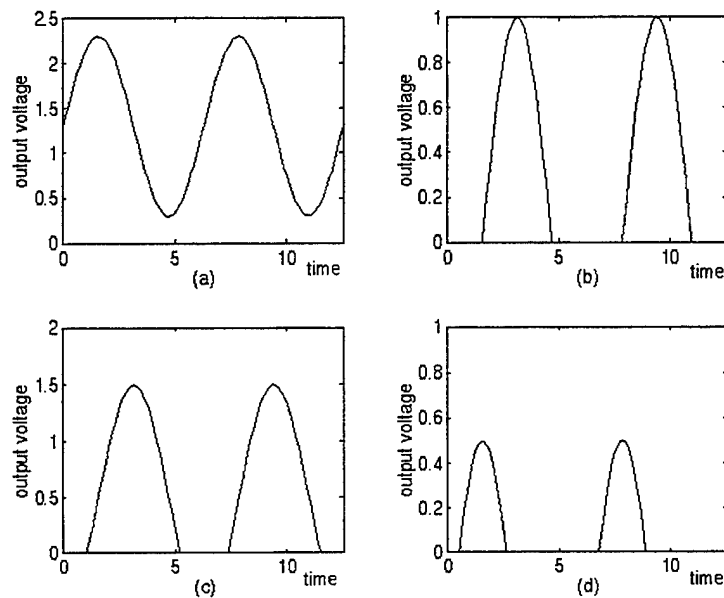


Figure 2.1: Amplifier Output Waveform Characteristics: (a) Class A amplifier, (b) Class AB amplifier, (c) Class B amplifier, (d) Class C amplifier.

## 5. High Efficiency Amplifiers.

There are many classes of amplifiers which operate at higher efficiencies than Class C amplifiers. The power efficiency is defined as the ratio of the output power to the collector's dc input power [Ref. 6]. These include Classes D, E, F, G, H, and S. The active devices in these amplifiers are transistors, in contrast to other classes which may use transistors *or* vacuum tubes. The first group of amplifiers (Class D, F, and S) employs the transistors as switches, not current sources, where the switch is on for half the cycle and off for the other half. The resulting waveform is a square wave with the design goal of minimizing the switching time. Minimizing the switching time, in turn, minimizes the transmission power loss. The square wave is then filtered to produce a sine wave. These amplifiers do not employ vacuum tubes as switches since the plate-cathode voltage drop cannot change sufficiently rapidly to produce a square wave [Ref. 6]. The second group of amplifiers (Classes E, G, and H) uses multiple power supply voltages, harmonic resonators,

and other circuit technologies. The overall advantage of high efficiency amplifiers is the ability to use smaller power supplies to generate the same output as other classes of amplifiers. These amplifiers can be found in cellular phone systems and other applications where limited power is available [Ref. 6].

## B. INTERMODULATION DISTORTION.

Intermodulation Distortion (IMD) is the nonlinear distortion produced as a result of the inability of an RF power amplifier to exactly reproduce the envelope and phase of an amplified signal. This distortion has many sources. They include: crossover effects, gain reduction at high current, variation of collector capacitance with collector voltage, and device saturation. This thesis considers the nonlinear distortion resulting from an HPA operating at its saturation point [Ref. 4].

The nonlinear distortion of an HPA operating at its saturation point has two components: Amplitude-to-Amplitude (AM-to-AM) conversion and Amplitude-to-Phase (AM-to-PM) conversion. AM-to-AM conversion is a result of the amplifier's inability to output an exact replica of the input waveform [Refs. 2 and 6]. AM-to-PM conversion results from the signal being passed through a reactance larger than the collector's load resistance causing a phase variation in the carrier frequency. AM-to-PM conversion is not detectable by an envelope detector but does cause unwanted sideband frequencies. The extent of this nonlinear distortion varies with the class, age, temperature, and composition of the HPA. As a result, many HPAs must be measured experimentally to obtain an accurate model of their nonlinear distortion characteristics. One significant exception to the need to model nonlinear distortion experimentally is the Traveling Wave Tube (TWT) HPA. An analytical model, developed by Saleh [Ref. 7], serves as a standard model for research into the design of predistorters and channel effects on signals amplified using a TWT. The AM-to-AM conversion is characterized by:

$$A(r) = \frac{\alpha_a r}{(1 + \beta_a r^2)} \quad (2.1)$$

The model for the AM-to-PM conversion is characterized by:

$$\Phi(r) = \frac{\alpha_\phi r^2}{(1 + \beta_\phi r^2)}, \quad (2.2)$$

where  $r$  is the input signal's amplitude. The coefficients  $\alpha_a$ ,  $\alpha_\phi$ ,  $\beta_a$ , and  $\beta_\phi$  are real and valued to approximate the standard TWT HPA distortion. The coefficients identifying the HPA in the communication system modeled in this thesis are obtained from Saleh. As the value of  $r$  becomes large, the AM-to-AM conversion may be approximated by  $1/r$  and the AM-to-PM conversion approaches a constant value. Figure 2.2 shows the change in non-linear distortion with respect to the magnitude of the signal amplitude [Ref. 7].

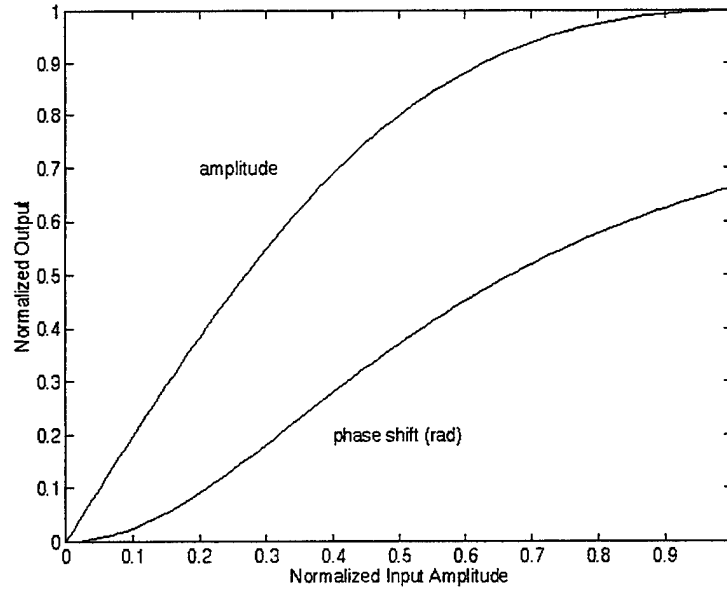


Figure 2.2: Nonlinear Distortion of TWT HPA.

One method of correcting the distortion of a TWT HPA is to operate it at a lower power level. This requires the amplifier to operate away from the saturation point, lowering its power efficiency. A major problem with the HPA's distortion characteristics is that as the sophistication of the signal constellation increases, increasing the need for linearity, the



HPA must be “backed off” further from the saturation point to maintain an acceptable bit error rate (BER). As a result, M-PSK has been used as a modulation scheme rather than M-QAM due to the less severe distortion of the signal constellation and the ease of adjusting the detection scheme to correctly decode symbols. “Backing off” the power level not only lowers the efficiency of the HPA, but also decreases the transmission range of the communication system. Alternate means of correcting this distortion must be found to allow the HPA to operate closer to its saturation point, increasing its efficiency and transmission range [Refs. 8 and 9].

### C. INTERMODULATION DISTORTION CORRECTION.

There are two methods to correct the intermodulation distortion in a TWT HPA. The first is to “back off” the transmitted power of the amplifier, decreasing the power efficiency to operate in the amplifier’s linear region. “Backing off” is used when no means of linearizing the HPA exists, but this is not a preferred means of linearization due to the reduced output power and efficiency. The second method to correct intermodulation distortion requires finding efficient methods to linearize nonlinear power amplifiers, like the TWT HPA. These techniques focus on allowing the HPA to transmit at a power level as close to the saturation region as possible while reducing the probability of symbol error. Some of these techniques correct only the AM-to-AM distortion, not the entire HPA distortion. Hoile [Ref. 10] and Hayashei [Ref. 11] are among the many researchers who have utilized S parameters to correct the nonlinearities in power amplifiers. The S parameter methods focus mainly at the active device level and use small signal parameters. As a result, they do not always correctly model the characteristics of high power amplifiers. Others have used techniques such as decision-feedback equalizers or digital table lookup techniques to interpret symbols [Ref. 12].

A major area of current research interest is the incorporation of a predistorter into the communication system prior to the up converter and the high power amplifier stages. Analog predistorters using discrete devices in many forms have been proposed in the literature [Ref. 13]. Digitally predistorting the signal at baseband eases the hardware requirements to perform the nonlinearity compensation. All predistorter models focus on

determining an inverse model of the HPA's AM-to-AM distortion effects while others focus on both the AM-to-AM and AM-to-PM distortion effects. These inverse models have been developed using neural network [Ref. 13], polynomial [Ref. 14], and adaptive filter approximations [Ref. 15]. The use of Volterra series approximations to develop an inverse model has been discussed in the literature [Ref. 16]. This thesis focuses on using Volterra series to determine an inverse model for inclusion as a predistorter into a communication system. Two methods are used to determine the inverse of the HPA's distortion characteristics. An MMSE method is used to provide Volterra approximations of higher nonlinearity orders than the Volterra approximations of other researchers [Refs. 8, 9, and 16]. The MMSE method uses complete kernel coefficient models, as well as limited kernel coefficient models, to represent the inverse of the distortion characteristics. The second method is an adaptive Recursive Least Squares (RLS) model. The RLS method uses the Volterra kernel coefficients to represent the inverse model's parameters.



### III. DIGITAL MODULATION

This chapter discusses the digital transmission of information. The data (binary) sequence is segmented into  $L$ -bit words which are mapped to unique points of a modulation scheme's signal constellation. This study focuses on Quadrature Amplitude Modulation (QAM) which uses voltage levels of two orthogonal signals to represent an  $L$  bit word. Noise introduced into the modulation process may degrade the original signal to a potentially undetectable signal. The goal is to design the transmitter and receiver in such a way as to minimize the number of errors in the received signal.

#### A. QUADRATURE AMPLITUDE MODULATION.

Modern communication systems are consuming ever increasing amounts of the radio frequency bandwidth. The information age requires that large quantities of information be transmitted at higher bit rates, which increases bandwidth requirements when using existing modulation schemes. Engineers designing communication systems are attempting to transmit the data with increasingly sophisticated modulation schemes to increase bandwidth efficiency. These sophisticated modulation schemes transmit a greater amount of information over the same or less bandwidth than simpler modulation schemes. A popular scheme of interest is  $M$ -Quadrature Amplitude Modulation ( $M$ -QAM), where  $M=2^L$ ,  $L = 1, 2, 3, 4, \dots$ , representing the number of bits/symbol. The higher the value of  $M$  the greater the number of bits an  $M$ -QAM modulation scheme may transmit in each symbol. This thesis considers 64-QAM, where  $L = 6$  bits of information per symbol. The general QAM signal is represented by

$$s(t) = A_I g(t) \cos(2\pi f_c t) - A_Q g(t) \sin(2\pi f_c t) \quad (3.1)$$

where  $A_I$  and  $A_Q$  are the amplitude values of the inphase and quadrature phase signals,  $g(t)$  is the pulse shaping signal, and  $f_c$  is the carrier frequency of the signal.

The amplitude values  $A_I$  and  $A_Q$  are used to form signal constellations in the I-Q plane. Each of the signal constellation's points has a unique representation of  $L$  bits which

uses the first  $L/2$  bits to represent the  $A_I$  coordinate and the second  $L/2$  bits to represent the  $A_Q$  coordinate. These  $L$  bits build one symbol. This reduces the number of units required to represent the same amount of information by a factor of  $L$ . This in turn allows the symbol data rate,  $R_s$ , to be lower than the bit rate,  $R_b$ , by a factor of  $L$ . This reduction in the symbol rate decreases the amount of RF bandwidth required to send the data, increasing the bandwidth efficiency of a communication system.

The signal constellation for a 64-QAM signal is shown in Figure 3.1. The convention used in this thesis represents the inphase plane component along the x (horizontal) axis and quadrature component along the y (vertical) axis [Ref. 17].

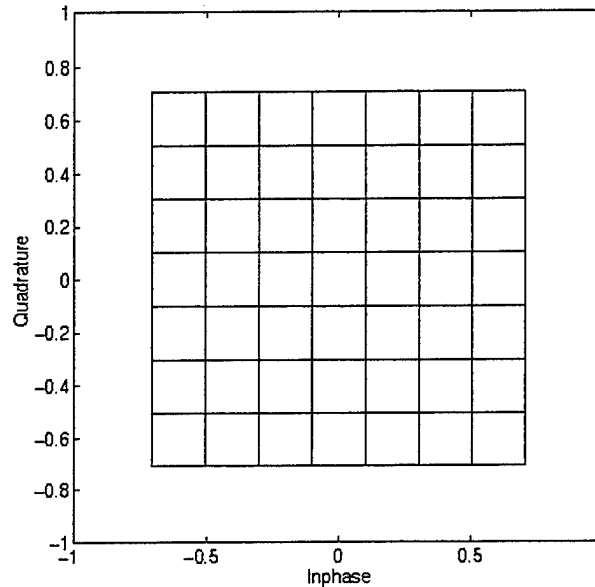


Figure 3.1: Original 64-QAM Signal Constellation.

## B. QUADRATURE AMPLITUDE MODULATION TRANSMITTER.

The QAM transmitter modeled in this thesis is shown in Figure 3.2. A table lookup function is performed by the logic device which reads the  $L$  bits and identifies the corresponding unique point in the QAM signal constellation. The QAM symbol, in complex form, is divided into its real and imaginary parts. The appropriate amplitude values are then forwarded to the upconversion stage of the modulator. The real (inphase)

part of the signal enters the top branch and the imaginary (quadrature) part of the signal enters the lower branch. Each branch shapes the amplitude value with the pulse generator's rectangular signal pulse. The rectangular pulses are then mixed with the carrier signals at center frequency  $f_c$ . The two modulated signals, orthogonal to each other, are then summed to obtain a real valued signal. The resulting signal is then amplified by the RF HPA to produce the signal power required to transmit the symbol to the receiver.

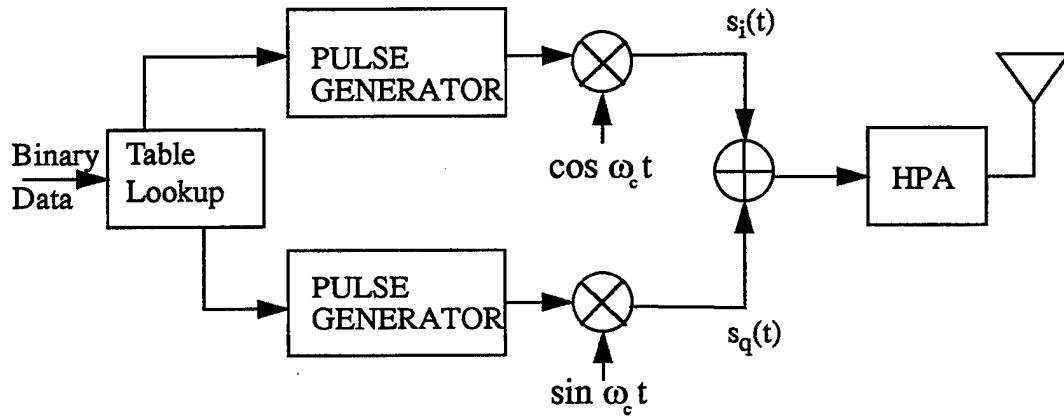


Figure 3.2: 64-QAM Modulator.

### C. QUADRATURE AMPLITUDE MODULATION RECEIVER.

The QAM receiver model is shown in Figure 3.3. The receiver is similar to other receivers using orthogonal signalling. The incoming signal is processed by both branches of the QAM receiver where the mixers multiply the signal by the appropriate carrier signals. The output of each mixer is the appropriate inphase or quadrature signal containing baseband and  $2f_c$  components. The output of the inphase mixer is

$$r_i(t) = A_I \left[ \frac{1}{2} + \frac{1}{2} \cos(4\pi f_c t) \right] \quad (3.2)$$

and the output of the quadrature mixer is

$$r_q(t) = A_Q \left[ -\frac{1}{2} + \frac{1}{2} \cos(4\pi f_c t) \right] \quad (3.3)$$

where  $A_I$  is the inphase voltage level,  $A_Q$  is the quadrature voltage level, and  $f_c$  is the carrier frequency. The branch signals are then processed by an integrator or a summer in a digital system. The transfer function of the integrator is a low pass filter, which eliminates the  $2f_c$  component from the signal. The remainder of the signal, the baseband component, is summed over one symbol period,  $T_s$ . The output from each summer is then processed by a logic device to determine the received symbol and decode the output bits [Ref. 18].

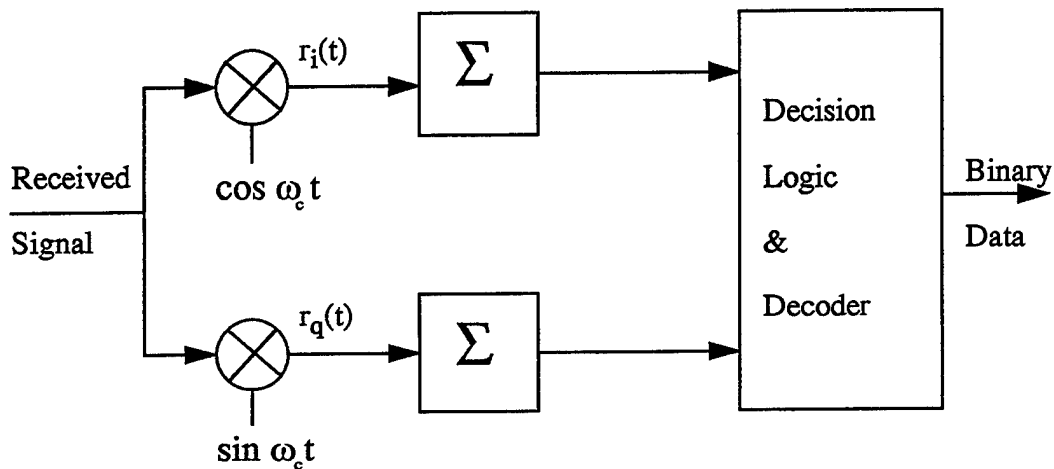


Figure 3.3: 64-QAM Receiver.

#### D. PROBABILITY OF SYMBOL ERROR.

The probability of symbol error for a QAM communication system is dependent upon the signal constellation used in the modulator. The quantity used in determining the probability of error is the minimum distance,  $d_{min}$ , between the signal constellation's adjacent points. The 64-QAM signal constellation employed here is rectangular,  $8 \times 8$ ; as a result each signal point has the same  $d_{min}$ . Rectangular signal constellations allow the communication system to easily decompose the QAM signal into two Pulse Amplitude Modulation (PAM) signals on two orthogonal carrier waveforms. The resulting rectangular constellation has an average power only slightly larger than the ideal signal constellation [Ref. 19]. The orthogonal signaling allows the QAM system's probability of error to be

derived from the PAM signal. The probability,  $P_c$ , that the transmitted symbol is decoded correctly is

$$P_c = (1 - P_8)^2 \quad (3.4)$$

where  $P_8$  is the probability of error for the 8-PAM communication system. The 8-PAM system possesses an average symbol power half that of the quadrature branches in the QAM system. The probability of error for an 8-PAM signal constellation is

$$P_8 = 2 \left(1 - \frac{1}{8}\right) Q \left( \sqrt{\frac{3}{63} \cdot \frac{E_{av}}{N_o}} \right) \quad (3.5)$$

where  $E_{av}$  is the average signal energy,  $N_o$  is the noise energy, and  $Q(\cdot)$  is the complementary error function. The final probability of symbol error,  $P_s$ , for QAM is found from the probability of a correct decision as  $P_s = 1 - P_c$  which yields  $P_s = 1 - (1 - P_8)^2$ .

Figure 3.4 shows the theoretical probability of symbol error for different levels of QAM signaling. It is apparent that as the number of bits/symbol ( $L$ ) increases the communication system requires a higher signal to noise ratio to maintain an equivalent probability of error. The performance of QAM is superior to comparable levels of M-Phase Shift Keying (M-PSK) when the nonlinear distortion of an HPA is absent [Refs. 17 and 19]. This provides the motivation to employ QAM as a bandwidth efficient modulation scheme. The probability of bit error for the M-QAM system under discussion is determined to be the symbol error rate divided by the number of bits per symbol. This approximation is made possible by the use of Gray coding. Errors usually occur on only one branch of the



demodulator and symbol decoder. This causes only one bit, out of the  $L$  symbol bits (6 in 64-QAM), to be in error [Ref. 19].

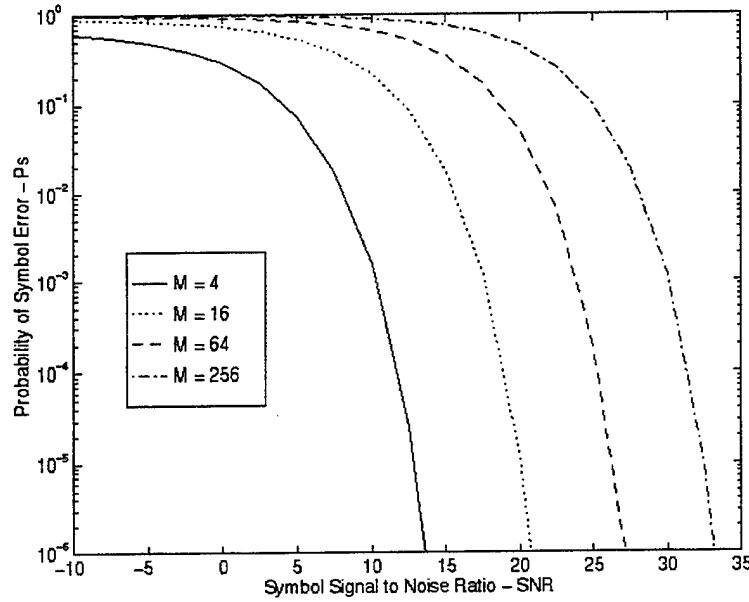


Figure 3.4: M-QAM Probability of Symbol Error vs. Symbol SNR.

#### E. HIGH POWER AMPLIFIER INFLUENCE.

The Intermodulation Distortion (IMD) introduced by the TWT HPA, operating near the saturation region, affects the amplitude and the phase components of the transmitted signal constellation. IMD is caused by Intersymbol Interference (ISI) and Adjacent Channel Interference (ACI) [Ref. 20]. The original 64-QAM constellation (see Figure 3.1) is an  $8 \times 8$  matrix with 64 distinct signal constellation points. Each point is assigned a unique 6-bit combination to represent one symbol. The 6-bit combinations are arranged in the signal constellation using a Gray code where only one bit changes between any two adjacent constellation points. This lowers the probability of bit error in the event that the receiver incorrectly decodes the received symbol [Ref. 17].

The signal emitted from the summer of the QAM transmitter is given by

$$x(t) = r(t) \cos(2\pi f_c t + \psi(t)) \quad (3.6)$$

where  $r(t)$  is the amplitude of the signal,  $f_c$  is the carrier frequency, and  $\psi(t)$  is the phase component. This waveform is input to the TWT HPA where the TWT HPA alters the QAM signal constellation. The resulting TWT HPA output is

$$y(t) = A[r(t)] \cos(2\pi f_c + \psi(t) + \Phi(r(t))) \quad (3.7)$$

where  $A[r(t)]$  represents the AM-to-AM distortion and  $\Phi(r(t))$  the AM-to-PM distortion. This nonlinear distortion causes the signal constellation to rotate counterclockwise and expand as  $r$  increases. The 64-QAM signal constellation output from the HPA is shown in Figure 3.5. The original signal constellation is shown in the dotted lines while the distorted constellation in the solid lines. The majority of the signal constellation points are no longer within their decision region. The demodulator will now be unable to correctly decode these symbols. As a result, the higher levels of QAM cannot be employed satisfactorily. The distorted signal constellation demonstrates the need to implement a predistortion scheme to correct the HPA's distortion and lower the probability of symbol error [Ref. 7].

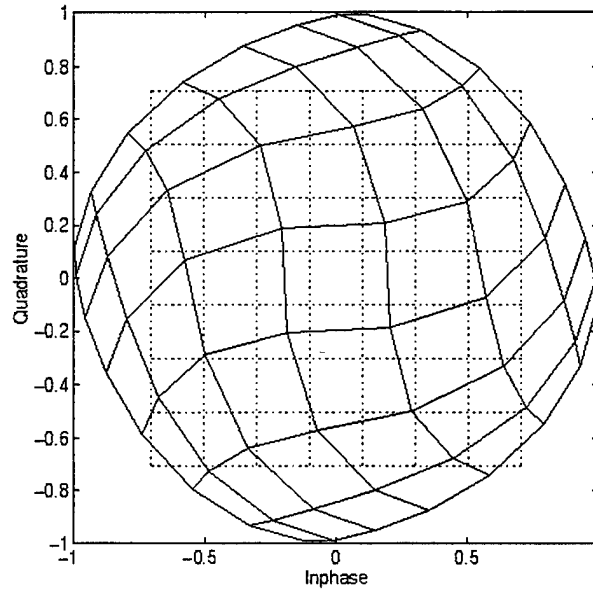


Figure 3.5: 64-QAM Distorted Signal Constellation.



## IV. VOLTERRA SERIES MODELING

This chapter discusses the modeling of nonlinear systems with the Volterra series. The Volterra series is essentially a Taylor series with memory, and is an excellent tool to model moderately nonlinear functions [Ref. 8]. A limited number of terms,  $p$ , in the Volterra series may be used to approximate the inverse of the TWT HPA's nonlinear distortion model proposed by Saleh [Ref. 7]. This approximation is the  $p$ th order inverse model and is used to design predistorters to decrease the probability of symbol error due to the nonlinear distortion of high power amplifiers.

### A. CONTINUOUS TIME VOLTERRA SERIES.

The Volterra series is a mathematical modeling technique used to model “black box” systems, which can be effective when a nonlinear relationship exists between the input and the output of the “black box.” The basic Volterra series system follows the form

$$y(t) = H[x(t)] \quad (4.1)$$

where  $H$  is the Volterra operator, which operates on an input  $x(t)$  to yield an output  $y(t)$ . The form of the operator  $H$  is an infinite series

$$H[x(t)] = H_1[x(t)] + H_2[x(t)] + \dots + H_n[x(t)] + \dots + H_\infty[x(t)] \quad (4.2)$$

where the  $n$ th order term has the form

$$H_n[x(t)] = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} h_n(\tau_1, \dots, \tau_n) x(t - \tau_1) \dots x(t - \tau_n) d\tau_1 \dots d\tau_n. \quad (4.3)$$

Thus the Volterra series representation of the system can be written explicitly as

$$\begin{aligned}
 y(t) = & h_0 + \int_{-\infty}^{\infty} h_1(\tau_1)x(t - \tau_1)d\tau_1 \\
 & + \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2)x(t - \tau_1)x(t - \tau_2)d\tau_1d\tau_2 \\
 & + \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_3(\tau_1, \tau_2, \tau_3)x(t - \tau_1)x(t - \tau_2)x(t - \tau_3)d\tau_1d\tau_2d\tau_3 + \dots
 \end{aligned} \tag{4.4}$$

In practice, it is not always necessary or possible to use an infinite number of terms to produce a sufficiently accurate approximation when using the Volterra series to model a system [Refs. 8 and 9]. In many cases it is possible to approximate the “black box” with sufficient accuracy using a finite number,  $p$ , of terms. The goal of this thesis is to determine the number of terms required to reach an accurate approximation of the nonlinear model. The systems considered in this research were time-invariant due to the memoryless nature of the high power amplifier’s response. Treating the systems as time-invariant also helps keep the problem’s complexity manageable [Refs. 21 and 22].

## B. DISCRETE TIME VOLTERRA SERIES.

In the discrete form, the Volterra series representation becomes

$$\begin{aligned}
 y(n) = & h_0 + \sum_{i_1=-\infty}^{\infty} h_1(n - i_1)x(i_1) \\
 & + \sum_{i_1=-\infty}^{\infty} \sum_{i_2=-\infty}^{\infty} h_2(n - i_1, n - i_2)x(i_1)x(i_2) \\
 & + \sum_{i_1=-\infty}^{\infty} \sum_{i_2=-\infty}^{\infty} \sum_{i_3=-\infty}^{\infty} h_3(n - i_1, n - i_2, n - i_3)x(i_1)x(i_2)x(i_3) + \dots
 \end{aligned} \tag{4.5}$$

where  $h_j$  is the function operator and  $x(i_n)$  is a data input term. The notation of the discrete time Volterra series can be simplified by substituting the Volterra operator, in the same

manner as with the continuous time Volterra series, to yield

$$y(n) = h_0 + H_1[x(n)] + H_2[x(n)] + \dots + H_n[x(n)] + \dots \quad (4.6)$$

where  $H_n[x(n)]$  is the Volterra operator. The discrete time Volterra series is solved in the least squares sense to obtain a finite solution using the matrix equation

$$\mathbf{y} = \mathbf{X}\mathbf{h} \quad (4.7)$$

where  $\mathbf{y}$  is a column vector of the HPA's output values,  $\mathbf{h}$  is a column vector with the Volterra kernel coefficients, and  $\mathbf{X}$  is a data matrix with the HPA's input values. The vector  $\mathbf{y}$  is given by

$$\mathbf{y} = \begin{bmatrix} y(n) \\ y(n+1) \\ y(n+2) \\ y(n+3) \\ y(n+4) \\ \vdots \\ \vdots \\ \vdots \\ y(n+q) \end{bmatrix} \quad (4.8)$$

where  $q$  is the memory length of the approximation. The column vector of Volterra kernel coefficients,  $\mathbf{h}$ , is given by

$$\mathbf{h} = \begin{bmatrix}
h_1(n - i_1 + 2) \\
h_1(n - i_1 + 1) \\
h_1(n - i_1) \\
h_2(n - i_1 + 2, n - i_2 + 2) \\
h_2(n - i_1 + 2, n - i_2 + 1) \\
h_2(n - i_1 + 1, n - i_2 + 2) \\
h_2(n - i_1 + 1, n - i_2 + 1) \\
h_2(n - i_1 + 2, n - i_2) \\
h_2(n - i_1, n - i_2 + 2) \\
h_2(n - i_1 + 1, n - i_2) \\
h_2(n - i_1, n - i_2 + 1) \\
h_2(n - i_1, n - i_2) \\
h_3(n - i_1 + 2, n - i_2 + 2, n - i_3 + 2) \\
h_3(n - i_1 + 2, n - i_2 + 1, n - i_3 + 1)
\end{bmatrix}. \quad (4.9)$$

The data matrix  $\mathbf{X}$  is constructed of the input signal data to the model. The size of  $\mathbf{X}$  depends upon the number of data points, the memory length of the system,  $q$ , and the order of the nonlinearity. The number of data points used to approximate the model determines the number of columns in the matrix. The number of rows depends on the memory and nonlinearity order of the series approximation. An example of a data matrix showing portions of the first three orders of nonlinearity with two delay terms is given by

$$\mathbf{X} = \begin{bmatrix}
x(n) & x(n+1) & x(n+2) \\
0 & x(n) & x(n+1) \\
0 & 0 & x(n) \\
x(n)x(n) & x(n+1)x(n+1) & x(n+2)x(n+2) \\
0 & x(n)x(n+1) & x(n+2)x(n+1) \\
0 & x(n+1)x(n) & x(n+1)x(n+2) \\
0 & x(n)x(n) & x(n+1)x(n+1) \\
0 & 0 & x(n+2)x(n) \\
0 & 0 & x(n)x(n+2) \\
0 & 0 & x(n+1)x(n) \\
0 & 0 & x(n)x(n+1) \\
0 & 0 & x(n)x(n) \\
x(n)x(n)x(n) & x(n+1)x(n+1)x(n+1) & x(n+2)x(n+2)x(n+2) \\
0 & x(n+1)x(n)x(n) & x(n+2)x(n+1)x(n+1)
\end{bmatrix}^H \quad (4.10)$$

where superscript  $H$  denotes the Hermetian transpose. As the order of nonlinearity and memory of the approximation are increased, additional rows can be added to the data matrix. Potentially, the accuracy of the series approximation will improve as the order of nonlinearity, as well as the memory length,  $q$ , are increased [Refs. 23 and 24].

### C. FINITE ( $p$ th) ORDER INVERSE.

The “black box” model to be determined here is the predistortion model used to represent the inverse of the distortion affects of the HPA. This requires finding an inverse of the Volterra series model of the HPA. The approximation of the HPA is a  $p$ th order model where  $p$  is the order of the nonlinearity. The inverse, if it exists, is the  $p$ th order inverse of the HPA model. The  $p$ th order inverse, when placed in cascade with the  $p$ th order model, should yield an impulse for the first order nonlinearity term and zero for the second through  $p$ th order terms. The nonlinear terms for orders  $p + 1$  and above may be nonzero. The cascade model is shown in Figure 4.1.



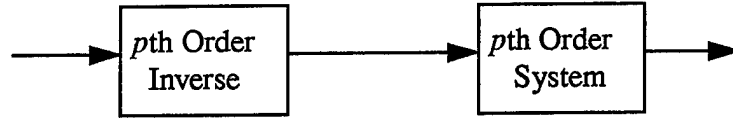


Figure 4.1: Volterra Series Cascade Model.

Not all Volterra series models possess an inverse. For a highly nonlinear system, it may be impossible to accurately model its inverse using a Volterra series. One alternative is to limit the operating range of the model, i.e., restrict operation to an area where the model is weakly nonlinear. Limiting the operating range may lead to the existence of a  $p$ th order inverse which can be applied to the model's operational range [Ref. 22].

#### D. SIMPLIFICATION OF THE VOLTERRA SERIES MODEL.

The traveling wave tube high power amplifier possesses properties which may be exploited to limit the size of the data matrix and decrease the number of calculations required. First, since the nonlinear system under consideration is bandpass in nature, all of the even terms in the Volterra series are zero [Refs. 8 and 9]. These terms can therefore be ignored and eliminated from the data matrix. Secondly, as mentioned earlier, the Volterra series can accurately model a nonlinear system with a limited number of terms. This lowers the order of the nonlinearity which must be used to calculate the predistorter. Thirdly, the Volterra series, for a time-invariant system, can be represented with symmetric kernels with no loss of generality [Ref. 22]. That is

$$h_2(n_1, n_2) = h_2(n_2, n_1) \quad (4.11)$$

and

$$h_3(n_1, n_2, n_3) = h_3(n_2, n_1, n_3) = \dots = h_3(n_3, n_2, n_1) \quad (4.12)$$

where  $h_2$  is a second order Volterra kernel coefficient,  $h_3$  is a third order Volterra kernel coefficient, and  $n_1$ ,  $n_2$  and  $n_3$  are time delays. The use of symmetric kernels throughout the modeling procedure simplifies the model and reduces the number of data terms in the data matrix [Ref. 25]. Finally, the amount of memory,  $q$ , in the system can be limited to keep the number of kernel coefficients in each nonlinearity order manageable. Models with long memories and several orders of nonlinearity require a large number of Volterra kernel coefficient terms. The foregoing simplifications allow the use of a much reduced data matrix to calculate the Volterra kernel coefficients. The updated data matrix for nonlinearity order  $p = 5$  and a memory of  $q = 2$  delays is shown in Equation 4.13. Using the smaller data matrix without the redundant symmetrical terms, the even order nonlinearity terms and limited memory terms allows the predistorter to be calculated quickly and may allow periodic updates while the communication system is in operation.

$$\mathbf{X} = \begin{bmatrix}
 x(n) & x(n+1) & x(n+2) \\
 0 & x(n) & x(n+1) \\
 0 & 0 & x(n) \\
 x(n)x(n)x(n) & x(n+1)x(n+1)x(n+1) & x(n+2)x(n+2)x(n+2) \\
 0 & x(n)x(n+1)x(n+1) & x(n+2)x(n+1)x(n+1) \\
 0 & x(n+1)x(n)x(n) & x(n+2)x(n+2)x(n+1) \\
 0 & x(n)x(n)x(n) & x(n+1)x(n+1)x(n+1) \\
 0 & 0 & x(n+2)x(n+2)x(n) \\
 0 & 0 & x(n+2)x(n)x(n) \\
 0 & 0 & x(n+1)x(n+1)x(n) \\
 0 & 0 & x(n)x(n)x(n+1) \\
 0 & 0 & x(n)x(n)x(n) \\
 0 & 0 & x(n+2)x(n+1)x(n) \\
 x(n)^5 & x(n+1)^5 & x(n+2)^5 \\
 0 & x(n+1)^4 x(n) & x(n+2)^4 x(n+1)
 \end{bmatrix}^H \quad (4.13)$$

### E. FIXED DATA PREDISTORTER.

The predistorter is determined by solving for the  $p$ th order inverse of the HPA model. The distorted output data emitted by the HPA is used to build the data matrix. The basic matrix equation to be solved is similar to the matrix equation discussed previously

$$x = Yg \quad (4.14)$$

where  $x$  is a column vector of input data to the HPA,  $g$  is a column vector of the Volterra kernel coefficients of the inverse model, and  $Y$  is a matrix of the output of the HPA (but the input to the model). We need to determine  $g$ . Figure 4.2 shows the block schematic used to develop the inverse model of the HPA's distortion characteristics.

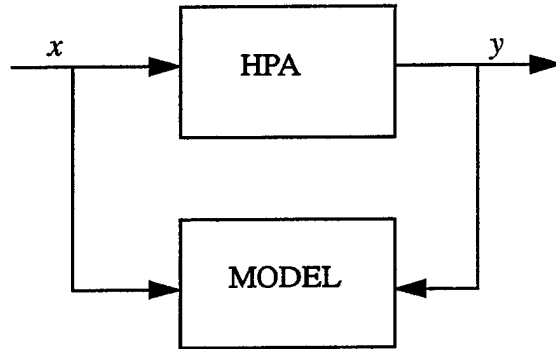


Figure 4.2: High Power Amplifier Model.

The Volterra kernel coefficients for the predistorter are determined by using the pseudoinverse approach [Ref. 27]. Reordering the equation to solve for the column vector,  $g$ , consisting of the Volterra kernel coefficients

$$\mathbf{g} = \mathbf{Y}^+ \mathbf{x} \quad (4.15)$$

where  $\mathbf{Y}$  is the data matrix,  $\mathbf{Y}^+$  is its pseudoinverse, and  $\mathbf{x}$  is the data vector. The data vector is defined by Equation 4.8. The Volterra kernel coefficient vector describes the  $p$ th order inverse model of the HPA and has the form

$$\mathbf{g} = \begin{bmatrix} g_1(n - i_1 + 2) \\ g_1(n - i_1 + 1) \\ g_1(n - i_1) \\ g_2(n - i_1 + 2, n - i_2 + 2) \\ g_2(n - i_1 + 2, n - i_2 + 1) \\ g_2(n - i_1 + 1, n - i_2 + 2) \\ g_2(n - i_1 + 1, n - i_2 + 1) \\ g_2(n - i_1 + 2, n - i_2) \\ g_2(n - i_1, n - i_2 + 2) \\ g_2(n - i_1 + 1, n - i_2) \\ g_2(n - i_1, n - i_2 + 1) \\ g_2(n - i_1, n - i_2) \\ g_3(n - i_1 + 2, n - i_2 + 2, n - i_3 + 2) \\ g_3(n - i_1 + 2, n - i_2 + 1, n - i_3 + 1) \end{bmatrix} \quad (4.16)$$

where  $g_i$  is a Volterra kernel coefficient. Finally, the data matrix  $\mathbf{Y}$  is given by

$$\mathbf{Y} = \begin{bmatrix}
y(n) & y(n+1) & y(n+2) \\
0 & y(n) & y(n+1) \\
0 & 0 & y(n) \\
y(n)y(n)y(n) & y(n+1)y(n+1)y(n+1) & y(n+2)y(n+2)y(n+2) \\
0 & y(n)y(n+1)y(n+1) & y(n+2)y(n+1)y(n+1) \\
0 & y(n+1)y(n)y(n) & y(n+2)y(n+2)y(n+1) \\
0 & y(n)y(n)y(n) & y(n+1)y(n+1)y(n+1) \\
0 & 0 & y(n+2)y(n+2)y(n) \\
0 & 0 & y(n+2)y(n)y(n) \\
0 & 0 & y(n+1)y(n+1)y(n) \\
0 & 0 & y(n)y(n)y(n+1) \\
0 & 0 & y(n)y(n)y(n) \\
0 & 0 & y(n+2)y(n+1)y(n) \\
y(n)^5 & y(n+1)^5 & y(n+2)^5 \\
0 & y(n+1)^4 y(n) & y(n+2)^4 y(n+1)
\end{bmatrix}^H \quad (4.17)$$

This forms the model for the predistorter. The coefficients  $g_i$  are then used to realize the predistorter. The modulation scheme's signal constellation symbol values are processed through the inverse model to determine the predistorted signal constellation. This constellation can then be used in a table lookup logic circuit to predistort incoming data points in a QAM system.

#### F. ADAPTIVE FILTER PREDISTORTER.

A second method of determining a predistorter is to use a recursive least squares (RLS) adaptive filtering technique. This technique allows periodic updating of the predistorter parameters and provides a partial solution for the parameters before a fully converged solution is obtained. Changes in the operational characteristics of the amplifier due to temperature, aging of the components, damage or defects in the device, and variations in the active device's operating point can be compensated for by the adaptive algorithm [Refs. 2 and 6]. The fixed Volterra series solution discussed in the Section E

calculates the predistorter using Saleh's model of the TWT HPA's distortion characteristics as constant. The fixed method does not update the model once it is installed in the communication system. The performance of the fixed predistorter may suffer if the TWT HPA's operating characteristics change during its operational life. The adaptive filter overcomes this problem.

The frequency of updating the adaptive algorithm can be varied depending upon the nature of the device, desired data rate, and the speed of the processor used to calculate the predistorter updates. In principle, the predistorter can be updated continuously, however, the time required to update the predistorter can limit the transmission speed if the predistorter is updated with each point transmitted. To prevent limiting the transmission speed, the initial predistorter is periodically updated off-line and the coefficients are inserted into the communication system upon completion of the predistorter's update.

### 1. Recursive Least Squares Estimation.

The RLS algorithm updates the tap weights, or filter parameters, recursively as each data symbol is received into the model. The goal of the algorithm is to minimize the error,  $e(n)$ , between the estimated signal output,  $\hat{x}(n)$ , and the actual signal output,  $x(n)$  [Ref. 28]

$$e(n) = x(n) - \hat{x}(n) = x(n) - \mathbf{g}^H(n)\mathbf{y}(n) \quad (4.18)$$

where  $\mathbf{g}(n)$  is a column vector of the Volterra kernel coefficients of the inverse model parameters and  $\mathbf{y}(n)$  is a column vector of HPA output data. The cost function to be minimized has the form

$$\varepsilon(n) = \sum_{n=0}^N \lambda^{N-n} e(n)e^*(n) \quad (4.19)$$

where  $\lambda^{N-n}$  is an exponential weighting function, and  $\lambda$  is referred to as the fading factor. The minimization requires that

$$\mathbf{R}(n)\mathbf{g}(n) = \mathbf{r}(n) \quad (4.20)$$

where  $\mathbf{R}(n)$  is the correlation matrix of the input data and  $\mathbf{r}(n)$  is the correlation vector [Ref. 28]. The RLS algorithm uses the matrix inversion lemma to represent the inverse correlation matrix  $\mathbf{P}(n) = \mathbf{R}^{-1}(n)$  as

$$\mathbf{P}(n) = \frac{1}{\lambda} \left[ \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\mathbf{y}(n)\mathbf{y}^H(n)\mathbf{P}(n-1)}{\lambda + \mathbf{y}^H(n)\mathbf{P}(n-1)\mathbf{y}(n)} \right] \quad (4.21)$$

where  $\mathbf{P}(n-1)$  is the inverse correlation matrix of the previous time sample and  $\mathbf{y}(n)$  is the data vector defined by Equation 4.25. The RLS algorithm is implemented as follows. The Kalman gain vector is defined as

$$\mathbf{k}(n) = \frac{\mathbf{P}(n-1)\mathbf{y}(n)}{\lambda + \mathbf{y}^H(n)\mathbf{P}(n-1)\mathbf{y}(n)} = \mathbf{P}(n)\mathbf{y}(n) \quad (4.22)$$

This recursively calculated Kalman gain vector is used to solve the inverse model as

$$\mathbf{g}(n) = \mathbf{g}(n-1) - \mathbf{k}(n)e(n) \quad (4.23)$$

where  $e(n)$  is the *a posteriori* error given by

$$e(n) = x(n) - \mathbf{y}^H \mathbf{g}(n-1) \quad (4.24)$$

This error or its squared value is used to determine the convergence behavior of the RLS algorithm. An example of the convergence of the RLS algorithm toward a predistorter solution is shown in the learning curve in Figure 4.3. This learning curve, using a

( $p=7, q=1$ ) Volterra series approximation, uses the mean-square error between the original signal constellation and the signal constellation emitted by the TWT HPA with predistortion. The mean-square error of the learning curve quickly decreases to a value close to that of the fixed data predistortion method after 1000 symbols [Refs. 19 and 28].

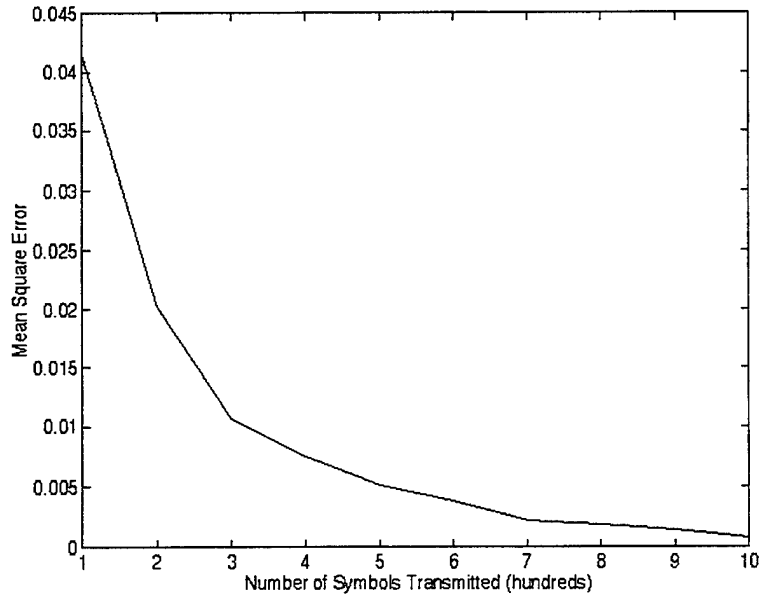


Figure 4.3: Learning Curve of ( $p=7, q=1$ ) Adaptive Predistorter, 1000 Symbols.

## 2. Adaptive Predistorter.

The adaptive RLS algorithm is employed to find the inverse of the TWT HPA's distortion characteristics, i.e., to identify the predistorter. The predistorter model's input vector,  $y(n)$ , consists of the odd order Volterra kernel coefficient terms of the model. This vector, for a fifth order nonlinearity with one delay, is given by



$$\mathbf{y}(n) = \begin{bmatrix} y(n+1) \\ y(n) \\ y^3(n+1) \\ y^2(n+1)y(n) \\ y(n+1)y^2(n) \\ y^3(n) \\ y^5(n+1) \\ y^4(n+1)y(n) \\ y^3(n+1)y^2(n) \\ y^2(n+1)y^3(n) \\ y(n+1)y^4(n) \\ y^5(n) \end{bmatrix} \quad (4.25)$$

and must be updated with each symbol transmitted by the communication system. This vector is used in Equations 4.21 through 4.23 until the inverse model produces a predistorter which achieves a desired *a posteriori* error or until a specified number of symbols has been transmitted. The final model parameter vector,  $\mathbf{g}(n)$ , is employed in a Volterra series approximation to generate the predistorter. An example of the model parameter vector,  $\mathbf{g}(n)$  with a 5th order nonlinearity and 1 memory delay is

$$\mathbf{g}(n) = \begin{bmatrix}
g_1(n - i_1) \\
g_1(n - i_1 + 1) \\
g_3(n - i_1, n - i_2, n - i_3) \\
g_3(n - i_1 + 1, n - i_2, n - i_3) \\
g_3(n - i_1 + 1, n - i_2 + 1, n - i_3) \\
g_3(n - i_1 + 1, n - i_2 + 1, n - i_3 + 1) \\
g_5(n - i_1, n - i_2, n - i_3, n - i_4, n - i_5) \\
g_5(n - i_1 + 1, n - i_2, n - i_3, n - i_4, n - i_5) \\
g_5(n - i_1 + 1, n - i_2 + 1, n - i_3, n - i_4, n - i_5) \\
g_5(n - i_1 + 1, n - i_2 + 1, n - i_3 + 1, n - i_4, n - i_5) \\
g_5(n - i_1 + 1, n - i_2 + 1, n - i_3 + 1, n - i_4 + 1, n - i_5) \\
g_5(n - i_1 + 1, n - i_2 + 1, n - i_3 + 1, n - i_4 + 1, n - i_5 + 1)
\end{bmatrix} \quad (4.26)$$

These inverse model parameter coefficients are then used to determine the predistorter signal constellation points. The predistorted signal constellation is then implemented in a table look up logic function.



## V. SIMULATION RESULTS

The preceding chapters deal with the predistorter models and the communication systems. This chapter focuses on testing the ability of these techniques to determine the inverse model of the TWT HPA's nonlinear characteristics, and implement the inverse model as a predistorter in a digital communication system. The results of these simulations are contained in the following sections.

### A. FIXED PREDISTORTER.

The fixed predistorter was implemented in MATLAB using the algorithm discussed in Section IV.E. The algorithm builds the data matrix using the HPA's output signal as the model's input in an iterative fashion i.e., a row of the Volterra model's data matrix with each iteration of symbol's transmission. The symbol is shifted into the memory buffer and the nonlinear data point coefficients are determined using the MATLAB function "order7.m" (Appendix A contains this function). The length of the vector produced by the function is dependent upon the number of delays incorporated into the model. The length of the output vector grows significantly as the nonlinearity order and the memory of the Volterra series model increases. Table 5.1 shows the number of symmetric terms contained in the output and Volterra coefficient vectors. Clearly, the coefficients of Volterra approximations with large nonlinearity orders and/or memory delays quickly become unmanageable. Therefore it is necessary to choose a low order, short memory approximation of the inverse of the distortion characteristics. This low order model will be used to realize a predistorter that hopefully results in an HPA emitted signal constellation that closely reproduces the original signal constellation, while retaining a manageable number of calculations. Simulation results for other problems indicate reasonable approximations can be obtained with a limited coefficient Volterra series approximation [Refs. 8, 13, 23, and 24].

Various predistorter models were determined for different nonlinearity orders and memory delays. The models will be referred to as  $(p,q)$ , where  $p$  is the nonlinear order of the Volterra series approximation, and  $q$  is the number of memory delays.

**Table 5.1: Number of Volterra Coefficients**

System Order	Memory Delays	No. of Terms
5	1	18
5	2	73
5	3	224
5	4	565
7	1	38
7	2	223
7	3	924

The modeling of the inverse characteristics was investigated by varying the nonlinearity order and keeping the memory constant at  $q=1$ . Each simulation used a data vector with 1000 symbols. The quantity used to judge the performance of the predistorter models is the mean-square error (MSE), defined as

$$MSE = \frac{\|d - d'\|^2}{64} \quad (5.1)$$

where  $d$  are the original signal constellation points and  $d'$  are the corrected signal constellation points emitted by the HPA with predistortion. The simulation results (see Figure 5.1) show that the Volterra series approximations reach a point of diminishing returns. The mean-square error improves as the nonlinearity order of the model increases from  $p=1$  to  $p=7$ , but reaches a minimum at orders  $p=7$  and  $p=9$ , and then increases with larger orders of nonlinearity. The mean-square error may increase at this point due to the increasing number of Volterra coefficients required to represent the models of higher nonlinearity

orders. Since the number of symbol points used to determine the predistorter model remains constant at 1000 symbols and the number of terms in the Volterra coefficient vector increases as the nonlinearity order increases, the matrix approaches a condition of being underdetermined. The larger the nonlinearity order of the model, the closer the data matrix approaches the condition of full rank, followed by the condition of an underdetermined system. This prevents solving the set of equations in a least squares sense. Incorporating a larger number of symbol points in the data matrix tends to overcome the underdetermined matrix problem.

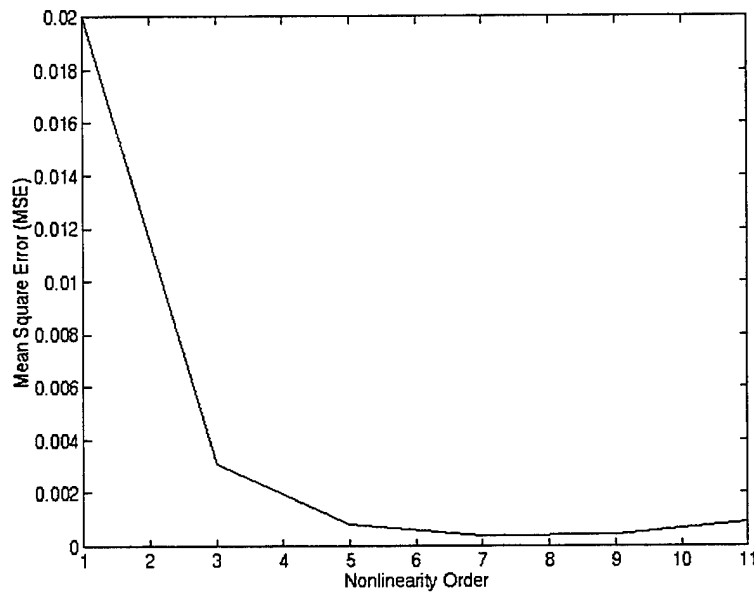


Figure 5.1: Fixed Predistorter Mean-Square Error, Memory Delay  $q = 1$ , versus Nonlinearity Order, 1000 data points.

The selection of the predistorter coefficients implemented in the communication system simulation was based upon the 5th and 7th order inverse models. Figure 5.2 depicts the predistorted signal constellation created by the fixed data method's Volterra coefficients for a  $(p=7, q=1)$  solution. Figure 5.3 depicts the transmitted signal constellation emitted by the HPA after the original signal constellation is predistorted by the inverse model of the TWT's nonlinear characteristics depicted in the signal constellation in Figure

5.2. The fixed data predistorter in Figure 5.2 results in a corrected signal constellation emitted by the HPA that has a MSE of 0.0002479.

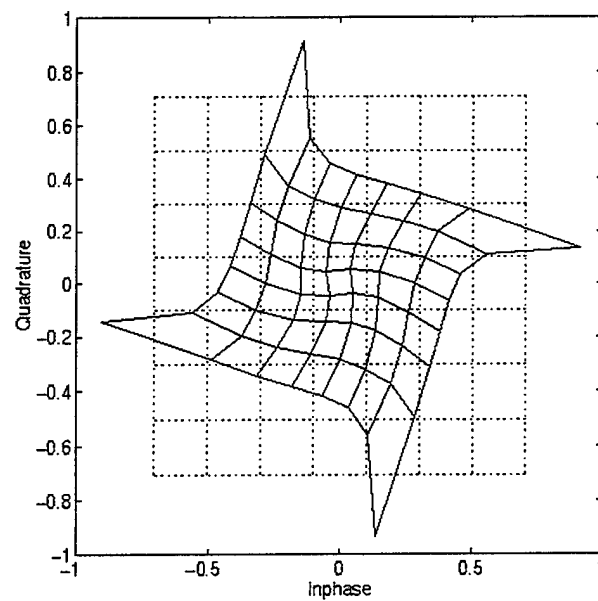


Figure 5.2: Fixed Predistorter, Nonlinearity Order  $p = 7$ , Memory  $q = 1$ .

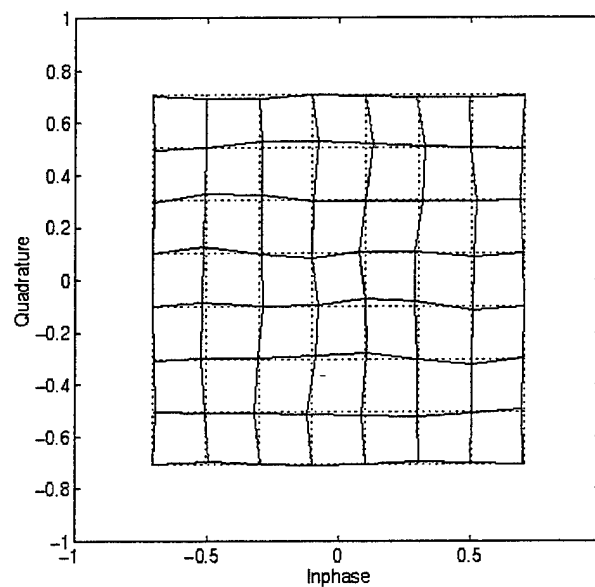


Figure 5.3: Corrected TWT HPA Signal Constellation Output with Predistortion

Even with a relatively low order and low memory, the number of coefficients and calculations required to solve a Volterra series approximation can get very large. Methods to further reduce the number of coefficients are essential to rapidly determine a predistorter model which can be employed in near real-time. Here we used a simple approach to reduce the number of coefficients. First the coefficients of a “full model” are determined. The magnitude of the coefficients are then plotted to determine their magnitude with respect to each other. The coefficients which have the greatest magnitude, therefore having the greatest impact on the approximation, are retained, and the remainder are deleted. This ensured that a  $p$ th order approximation had terms of the 1st, 3rd, 5th, ..., and  $p$ th order within the Volterra kernel. Figure 5.4 depicts a sample of the 5th order terms of a ( $p=7, q=1$ ) Volterra series approximation. The coefficients chosen to be retained in this “limited coefficient model” were  $x(1)x(1)x(1)x^*(1)x^*(1)$ ,  $x(1)x(1)x(2)x^*(1)x^*(2)$ , and  $x(1)x(2)x(2)x^*(1)x^*(1)$ . The coefficients which are retained are not known *a priori* and are selected only on the basis of their magnitude with respect to the other coefficients in the model. No analysis was performed to determine if these coefficients dominate the models in general.

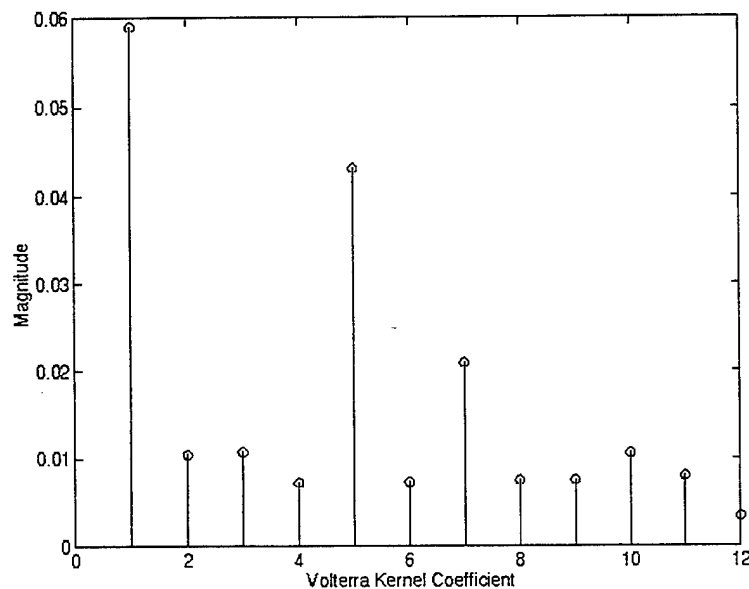


Figure 5.4: 5th Order Volterra Kernel Coefficients using a ( $p=7, q=1$ ) Fixed Volterra Series Predistorter.



The limited coefficient Volterra series approximation was used to form the limited coefficient predistorter model. This limited coefficient predistorter's signal constellation is shown in Figure 5.5. When this predistorted signal constellation is input into the TWT HPA, the resulting corrected signal constellation output from the TWT HPA is shown in Figure 5.6. This limited coefficient fixed data predistorter results in a corrected signal constellation emitted by the HPA that has a mean-square error of 0.0003946. This is close to the full coefficient solution, which has an MSE of 0.0002479. The limited coefficient predistorter model was therefore judged to be effective in implementing the predistortion with considerably less computation [Refs. 8 and 23].

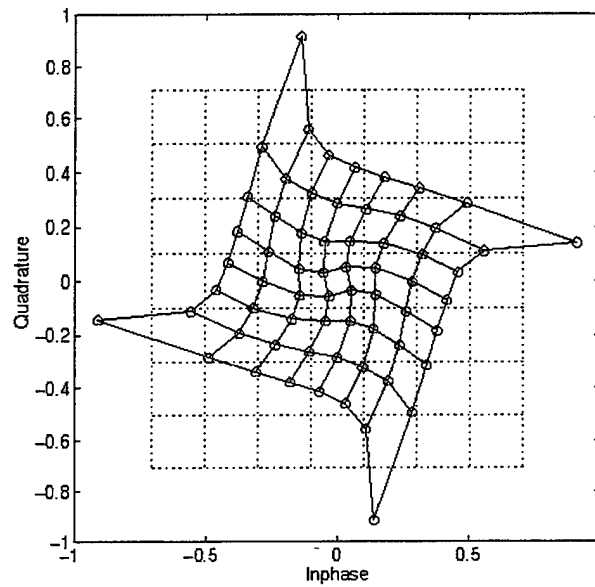


Figure 5.5: Fixed Predistorter, Nonlinearity Order  $p = 7$ , Memory  $q = 1$ , using the limited coefficient Volterra Approximation.

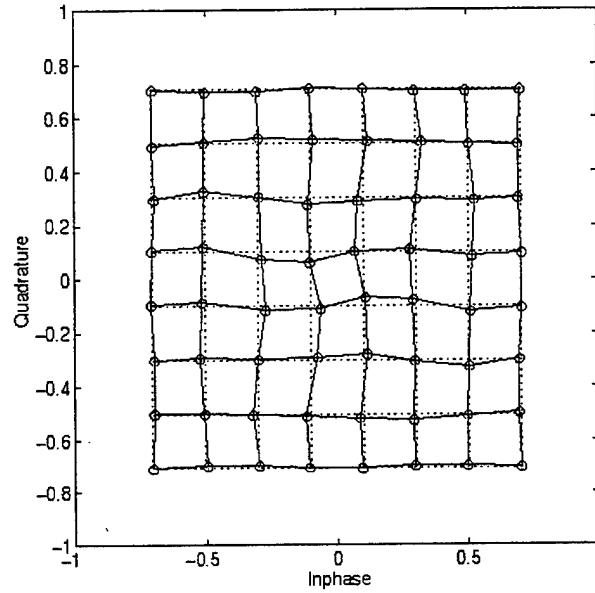


Figure 5.6: High Power Amplifier Output After Predistortion Using the Limited Coefficient ( $p=7, q=1$ ) Predistorter.

## B. ADAPTIVE PREDISTORTER.

The adaptive predistorter was determined using the RLS method discussed in Chapter IV. The algorithm was initialized with  $\lambda = 0.995$  and  $\delta = 15$  to allow rapid convergence. The predistorters generated by the RLS method have a similar signal constellation shape as the fixed predistorter model. The RLS algorithm required 500 to 1000 symbol points, depending on the memory and nonlinearity order of the system, to achieve a mean-square error similar to that of the fixed method. The convergence of the TWT HPA's emitted signal constellation, when predistorted with the inverse model determined with the RLS algorithm, is shown in the learning curve in Figure 4.3. The RLS algorithm was continued only for a sufficient number of symbol points to achieve a mean-square error comparable to that of the fixed data method. Figure 5.7 shows the predistorted signal constellation determined with an RLS ( $p=7, q=1$ ) Volterra approximation to model

the inverse nonlinear characteristics. This predistorted signal constellation is similar to the fixed predistorted signal constellation in Figure 5.3. When this predistorted signal constellation is input to the TWT HPA, the corrected signal constellation yields a mean-square error of 0.000554. The corrected signal constellation is shown in Figure 5.8.

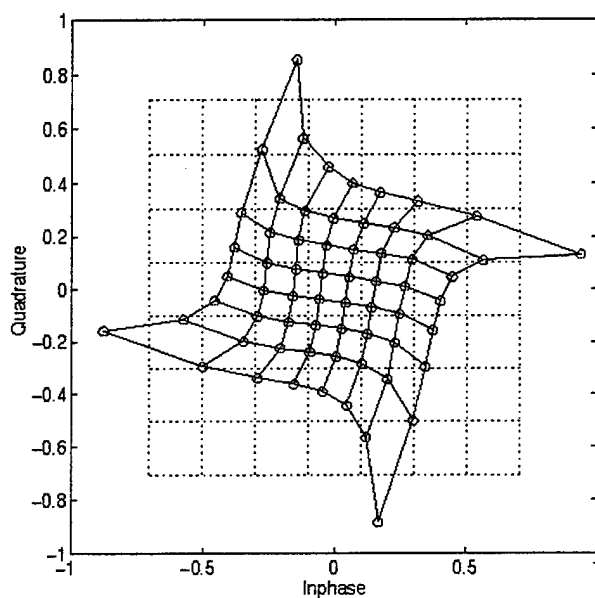


Figure 5.7: Adaptive Predistortion Signal Constellation using  $(p=7,q=1)$  RLS Predistorter.

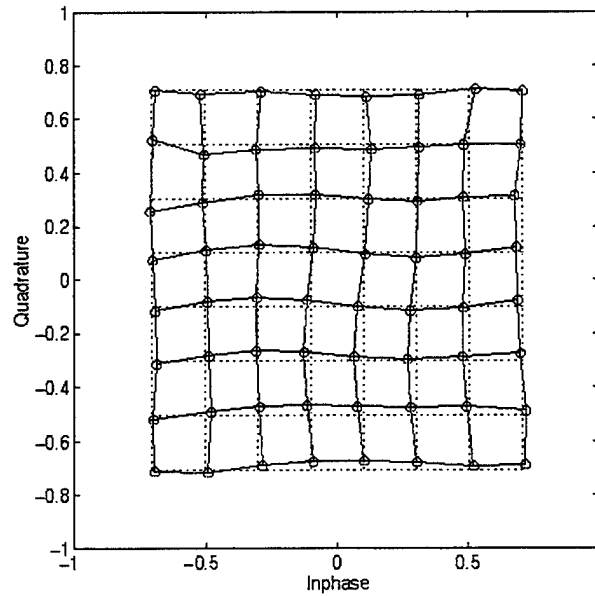


Figure 5.8: High Power Amplifier Output After Predistortion Using the Adaptive ( $p=7, q=1$ ) Predistorter.

## C. COMMUNICATION SYSTEM.

### 1. Modulator.

The modulator was designed to implement the quadrature amplitude modulation communication system discussed in Chapter III. The signal constellation point is predistorted in accordance with the predistorter model and prepared for transmission. The modulator (see Figure 5.9) employs a table lookup function to convert the  $L$  bits of binary data into a complex number representing a unique point in the signal constellation. Incorporating the predistorter model's signal constellation in the table lookup function eliminates the need to perform the predistortion as a separate step. A logic device breaks the signal constellation point into its real and imaginary parts, represented by

$$s_b(t) = A_I + i \cdot A_Q \quad (5.2)$$

where  $s_b(t)$  is the complex symbol,  $A_I$  is the predistorted voltage level assigned to the inphase bit stream, and  $A_Q$  is the predistorted voltage level assigned to the quadrature bit stream. The real component,  $A_I$ , is routed to the inphase branch and the imaginary component,  $A_Q$ , is sent to the quadrature branch of the modulator. The predistorted voltage levels are shaped by the pulse generator into a rectangular waveform to scale it to the transmitted voltage level. The inphase and quadrature signals after the pulse generators are

$$s_i(t) = A_I g(t - T) \quad (5.3)$$

and

$$s_q(t) = A_Q g(t - T) . \quad (5.4)$$

where  $g(t-T)$  is the rectangular pulse. The scaled rectangular pulse is then mixed with the carrier frequency resulting in the two modulated waveforms

$$s_{mi}(t) = A_I g(t - T) \cos(2\pi f_c t) \quad (5.5)$$

and

$$s_{mq}(t) = A_Q g(t - T) \sin(2\pi f_c t) . \quad (5.6)$$

The carrier frequency used in all of the simulations is 455 kHz, a common intermediate frequency in superheterodyne receivers. Each symbol was transmitted for 10 complete periods of 10 samples per period, i.e., a total of 100 samples per symbol. Additive white Gaussian noise (AWGN) was included in each channel to simulate the effect of thermal noise in the system. The output of each branch was then summed to produce a real valued output for transmission. The final waveform is

$$s(t) = A_I g(t - T) \cos(2\pi f_c t) - A_Q g(t - T) \sin(2\pi f_c t). \quad (5.7)$$

The HPA model was simulated using Saleh's model at baseband (see Section II.B) since the analytical expressions are only valid at baseband, not at the modulated RF frequencies. The HPA is placed after the predistorter as shown in Figure 5.9.

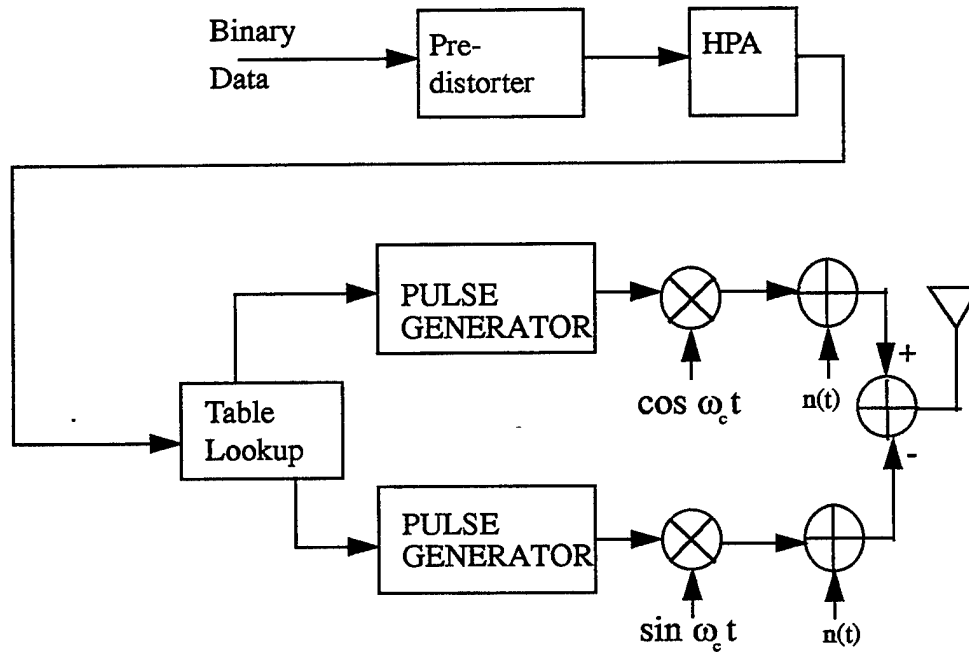


Figure 5.9: Simulated Communication System Modulator.

## 2. Demodulator.

The demodulator used in this simulation is greatly simplified from that of an actual implementation. The simplifications were made to focus the results on the effectiveness of the predistorter, not the demodulator. Several assumptions were made to achieve this simplification, including:

- a. No channel loss, distortions, or fading.
- b. Perfect synchronization of the modulator and the demodulator, in both phase and start of symbol transmission.
- c. No Noise (noise added to the system at the modulator only).

The demodulator is a standard quadrature digital demodulator (see Figure 3.3) designed to detect the received signal

$$r(t) = A'_I g(t - T) \cos(2\pi f_c t) - A'_Q g(t - T) \sin(2\pi f_c t) + n(t) \quad (5.8)$$

where  $A'_I$  is the received inphase voltage level,  $A'_Q$  is the received quadrature voltage level, and  $n(t)$  is additive noise. The received signal,  $r(t)$ , is input into both the inphase and quadrature branches and mixed with the local oscillator. The perfect synchronization assumption ensures that the signal output from the mixer consists of only the dc and the  $2f_c$  components of the modulated waveform, plus the noise injected into the system. The output of the branch mixers is

$$r'_i(t) = A'_I \left( \frac{1}{2} + \frac{1}{2} \cos(4\pi f_c t) \right) + n'_i(t) \quad (5.9)$$

and

$$r'_q(t) = A'_Q \left( -\frac{1}{2} + \frac{1}{2} \cos(4\pi f_c t) \right) + n'_q(t) \quad (5.10)$$

where  $n'_i(t)$  and  $n'_q(t)$  are the processed noise components. The signals of both branches are passed through a summing device, equivalent to an analog integrator, which has a transfer characteristic similar to that of a low pass filter. The summer outputs are sampled at the end of the symbol's transmission period as the received signal component. The branch summer outputs at the sampling time,  $t_o$ , are

$$r''_i(t_o) = \frac{A'_I}{2} + n''_i(t_o) \quad (5.11)$$

and

$$r''_q(t_o) = \frac{A'_Q}{2} + n''_q(t_o) \quad (5.12)$$

where  $n''_i(t_o)$  and  $n''_q(t_o)$  are the noise components. The summers of the two branches output the threshold levels to the detection logic to determine which symbol was received. The logic device chooses the closest table value to the transmission value for each branches. The received symbol is then created from the two branches and converted into its binary form. The  $L$  bits composing the symbol can then be serially output into the host receiver's system.

The symbol error rate (SER) is determined as follows for both fixed and adaptive predistortion techniques. The received symbol is compared with the transmitted symbol: if the received symbol does not match the transmitted symbol, it is considered an error. A running total of the errors is maintained. Upon completion of the simulation, the probability of symbol error,  $P_s$ , is estimated as

$$P_s = \frac{k}{N} \quad (5.13)$$



where  $k$  is the number of errors and  $N$  is the total number of symbols transmitted. The SERs were determined for a range of signal-to-noise ratios to obtain the system's performance trends in terms of the SER versus SNR curves and compare them to the theoretical curve for 64-QAM.

The difference between the two predistorters incorporated into the communication system is the updating of the inverse model in the adaptive method. The fixed predistorter communication system is not updated during its operation. The adaptive predistorter communication system can be updated continuously but in this thesis it is updated periodically, not continuously, to save calculation time. The update is performed by transmitting data in blocks of 10000 symbols each. The last 1000 symbols of the 10000 symbol blocks are stored in a buffer for use in updating the signal constellation of the predistorter off-line. When the last symbol of the block is transmitted, the adaptive RLS algorithm is employed to update the predistorter using the  $(p=7, q=1)$  inverse model with the buffer serving as the symbol data. The updated predistorter coefficients are then inserted into the communication system's  $(p=7, q=1)$  predistorter and the next block of symbols is transmitted.

### 3. Signal-to-Noise Ratio.

The signal-to-noise ratio (SNR) is determined by

$$SNR = \frac{P_{av}}{N_o} \quad (5.14)$$

where  $P_{av}$  is the average signal power

$$P_{av} = \frac{(A_I)^2 + (A_Q)^2}{2} \quad (5.15)$$

and  $N_o$  is the noise power. The noise power of the communication system is the sum of the

noise in the two branches of the modulator. The noise was added to each branch as close to the end of the modulator as possible, simulating the thermal noise in the system. Adding complex noise at baseband would be equivalent. The measured probability of symbol error versus SNR was used as the quantity to evaluate the performance of the communication system simulations. The MATLAB computer code implementing these simulations is in Appendix D and Appendix E. In these simulations, the SNR acted as a scale factor for the noise power generated by a random vector. This noise was added to the transmitted signal to determine the symbol detection and decoding performance of the communication system [Ref. 11].

#### D. COMMUNICATION SYSTEM PERFORMANCE.

The performance of the communication system in terms of error probability is shown in Figure 5.10. These simulations involved transmitting 100,000 symbols through the communication system. The ideal theoretical performance of 64-QAM is depicted by the solid curve highlighted by asterisks. This is the best possible decision performance a 64-QAM system may achieve under ideal conditions. A total of 1000 symbols were used to compute the inverse models to realize the fixed data predistorters. The fixed data predistorter systems with 7th order ( $p=7, q=1$ ) Volterra series approximations are the best performing systems analyzed in this thesis. As can be seen, the 7th order system ( $p=7, q=1$ ) with a complete set of Volterra kernel coefficients performs slightly better than the 7th order system with limited coefficients. The limited coefficient predistorter requires less than 0.5 dB more power than the complete coefficient fixed predistorter to achieve a symbol error probability of  $10^{-5}$ . This allows a slightly higher transmitting power to be traded off for a much reduced computation requirement. The 7th order ( $p=7, q=1$ ) adaptive predistorter required approximately 5dB more power than the ideal 64-QAM system, 2.5 dB more than the ( $p=7, q=1$ ) complete coefficient fixed data predistorter, to achieve a symbol error rate of  $10^{-5}$ . The adaptive algorithm used parameters of  $\delta=15$  and  $\lambda=0.995$  to initialize the algorithm and identify the memory dependence of the model. The RLS

method required approximately 1000 symbol iterations to produce an initial inverse model which yielded a corrected signal constellation with a mean-square error of 0.0007. Further updates to the predistorter were conducted as described in Section C.2. The 5th order ( $p=5, q=1$ ) predistorter demonstrated the poorest performance of the predistorters implemented. The ( $p=5, q=1$ ) predistorter required approximately 8.5 dB more power than the ideal 64-QAM system and 5 dB more power than the ( $p=7, q=1$ ) fixed data predistorters to achieve a symbol error rate of  $10^{-5}$ .

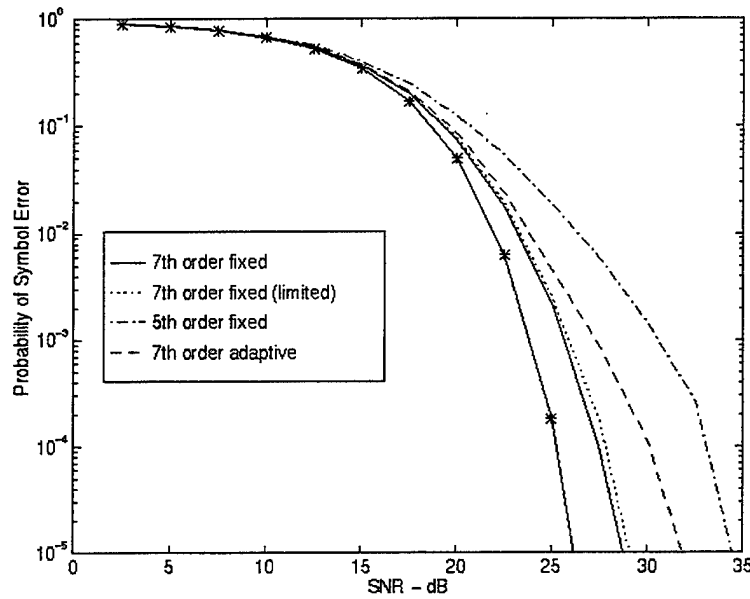


Figure 5.10: Communication System Performance; 100,000 Symbols Transmitted.

## VI. CONCLUSIONS

### A. CONCLUSIONS.

This thesis discusses two methods of determining a predistorter to correct the nonlinear distortion of the traveling wave tube high power amplifier. Specifically, Volterra series approximation techniques are developed using fixed and adaptive data modeling techniques. These techniques were used to determine an inverse model of the TWT HPA's nonlinear distortion characteristics which was then used to predistort the modulation scheme's signal constellation. The predistorted signal constellation formed the basis of a table lookup logic function for inclusion in the modulation scheme. The predistortion of the signal constellation's data symbols results in a transmitted signal that is similar to the original signal constellation's data points. The following conclusions can be made from the two predistortion techniques and their employment in the communication system simulation.

The Volterra series approximation techniques successfully modeled the inverse of the TWT's nonlinear distortion characteristics. The use of these inverse models as predistorters results in the high power amplifier's corrected output signal constellation being within the appropriate detection regions of the symbol decoder. The 7th order fixed data approximations generated the predistorter models with the best correction performance. The mean-square error between the 7th order fixed data approximation's corrected constellation and the original signal constellation was 0.000247. The 7th order adaptive approximation mean-square error was 0.000554 and the 5th order fixed data approximation was 0.000634.

The predistorters lowered the probability of symbol error in the 64-QAM communication system to levels comparable to the ideal 64-QAM communication system's performance. The 7th order fixed data predistorter required the lowest signal-to-noise ratio of the predistorters implemented in the 64-QAM communication system. The 7th order

fixed data predistorter required 2.5 dB more power than the ideal 64-QAM system to achieve a symbol error rate of  $10^{-5}$ . The 7th order adaptive predistorter required 2.5 dB and the 5th order fixed data predistorter 5 dB more power than the 7th order fixed data predistorter to achieve a symbol error rate of  $10^{-5}$ .

The failure to employ a predistortion method requires the receiver to decode the distorted signal. The conventional QAM receiver used in this manner proved incapable of achieving a satisfactory symbol error rate. The use of the predistorters would allow higher order modulation schemes to be employed in modern communication systems to transmit data at high transmission rates while conserving bandwidth.

## **B. FUTURE WORK.**

The results reported here provide a starting point for research into digital predistortion techniques to allow bandwidth efficient modulation schemes to be employed in high data rate wireless transmitters. The equations for the nonlinear distortion characteristics of the TWT HPA provide an excellent and recognized analytical model to determine the effects of HPAs on bandwidth efficient modulation schemes. The predistortion methods presented in this thesis are possible implementations to correct the nonlinear distortion of the 64-QAM signals.

There are many areas which provide excellent avenues for future investigation into predistortion techniques. One area is implementing the communication system without perfect synchronization to determine the effect of changing channel conditions on the predistorter. This will help determine if fading and other detrimental conditions cause the predistorters to be ineffective. A second area for future investigation is to implement the predistorters in C code. This will allow the predistorter to be operated in a 64-QAM communication system using digital signal processing (DSP) chips. The performance of the predistorters could then be evaluated under actual operating conditions for future fielding in communication systems.

## APPENDIX A. MATLAB COMPUTER CODE FOR DETERMINING THE NONLINEAR DATA COEFFICIENTS.

```
function [y,i1,i3,i5,i7] = order7(sr,m)

% function [y,i1,i3,i5,i7] = order7(sr,m)
%
% This function computes the first, third, fifth, and seventh
% order Volterra Series nonlinear data coefficients for a model.
% The coefficients computed do not duplicate symmetric
% coefficients, due to the model being time-invariant,
% to reduce the size of the resulting model and number
% of calculations required.
%
% Input: Shift register values and the model memory.
%
% Output: y, a column vector with the computed nonlinear data
% coefficients for the current shift register states and the
% number of data terms in the vector y. These are used in the
% script files which call order7.m.

% Initialize the counter for the number of terms.

i = 0;

% Calculate the first order terms.

for k = 1:m
    i = i+1;
    y(i,1) = sr(k);
end
i1=i;

% Calculate the third order terms.

for k = 1:m
    for l = k:m
        for n = l:m
            i = i+1;
            y(i,1) = sr(k)*sr(l)*conj(sr(n));
        end
    end
end
i3=i;
```

% Calculate the fifth order terms.

```
for k = 1:m
    for l = k:m
        for n = l:m
            for o = 1:m
                for p = o:m
                    i = i+1;
                    y(i,1) = sr(k)*sr(l)*sr(n)*conj(sr(o))*conj(sr(p));
                end
            end
        end
    end
end
i5=i;
```

% Calculate the seventh order terms.

```
for k1 = 1:m
    for k2 = k1:m
        for k3 = k2:m
            for k4 = k3:m
                for k5 = 1:m
                    for k6 = k5:m
                        for k7 = k6:m
                            i = i+1;
                            y(i,1) = sr(k1)*sr(k2)*sr(k3)*sr(k4)*conj(sr(k5))*conj(sr(k6))*conj(sr(k7));
                        end
                    end
                end
            end
        end
    end
end
i7=i;
```

## APPENDIX B. SUPPORTING MATLAB FUNCTIONS CALLED TO DETERMINE THE PREDISTORTER MODEL'S COEFFICIENTS.

### 1. FUNCTION HPA.M

```
function c2=hpa(c,ampparam,phparam);
```

```
% function c2=hpa(c,ampparam,phparam)
```

```
% This function computes the nonlinear distortion caused by the Travelling Wave Tube  
% high power amplifier. The function is complex and determines the Amplitude-to-Amplitude  
% (AM-to-AM) and Amplitude-to-Phase (AM-to-PM) distortion.
```

```
%
```

```
% Input: Complex symbol constellation point and high power amplifier characteristic  
% coefficients, if different from the default coefficients.
```

```
%
```

```
% Output: Distorted symbol constellation point. (Complex value)
```

```
%
```

```
% This function was written by LCDR Bruce Watkins, NCCOSC RDT&E DIV, San Diego,CA
```

```
r=abs(c);
```

```
a=angle(c);
```

```
% Default HPA characteristic coefficients.
```

```
a1=1.9945;
```

```
b1=0.9945;
```

```
a2=2.5293;
```

```
b2=2.8168;
```

```
if nargin==3
```

```
    a1=ampparam(1);
```

```
    a2=ampparam(2);
```

```
    a2=phparam(1);
```

```
    b2=phparam(2);
```

```
end
```

```
% Calculate the amplitude distortion.
```

```
A=a1*r./(1+b1*r.^2);
```

```
% Calculate the phase distortion.
```

```
P=a2*r.^2./(1+b2*r.^2);
```

```
% Increment the phase to reflect the phase distortion.
```



```
a=a+P;
```

```
% Determine the distorted symbol to be output.
```

```
c2=A.*cos(a)+i*A.*sin(a);
```

## 2. SCRIPT FILE ASPECT1.M.

```
% Script File aspect1.m.
```

```
% This script file changes the axes of figures to a square appearance for uniform plotting of  
% signal constellations. The script is called by both the fixed and adaptive predistorter  
% modelling techniques.
```

```
%
```

```
% Input: None.
```

```
%
```

```
% Output: None.
```

```
%
```

```
% This function was written by LCDR Bruce Watkins, NCCOSC RDT&E DIV, San Diego, CA
```

```
% this script sets the aspect ratio of the graphics figure to 1:1
```

```
%h1=[1 1];
```

```
axis([-1 1 -1 1])
```

```
axis('square')
```

```
%set(gca,'AspectRatio',h1);
```

## APPENDIX C. MATLAB COMPUTER CODE CALCULATING THE FIXED PREDISTORTER MODEL.

### 1. SCRIPT FILE F7.M

```
% f7.m
%
% This script file calculates the fixed predistorter model for a seventh order nonlinearity
% Volterra Series approximation. The script file serves mainly as a shell to call the functions
% which perform the calculations and output the predistorter model and the signal constellation
% emitted by the high power amplifier. The predistorter model determined is an inverse of
% the nonlinear distortion characteristics of the traveling wave tube high power amplifier.
%
% Input: None.
%
% Output: Figures with the predistorter model and the HPA's emitted signal constellation.

clear

% Identify the memory of the system. ord = memory delays + 1.

ord = 2;

% Load the Quadrature Amplitude Modulation signal constellation file and identify the
% constellation to be used.

load qam
xi = c64;

% Transform the signal constellation from a matrix to a vector.

x=reshape(xi,1,64);

% Identify the number of data points to be used to determine the predistorter model.

points = 1001;

% Call the function fix7.m. This function performs the actual calculations to determine the
% predistorter model.

[distdata,yv,g1,g3,g5,g7,i1,i3,i5,i7] = fix7(ord,x,points)

% "Transmit" the predistorter model through the high power amplifier.

yv = hpa(distdata);
% Reshape the vectors into 8 x 8 matrices for plotting.
```

```

yyv=reshape(yv,8,8);
dd=reshape(distdata,8,8);

% Calculate the mean square error of the resulting signal constellation emitted from the
% high power amplifier.

se = sum(sum(abs(yyv-conj(xi)).^2));sx=size(xi);
msefinal=se/(sx(1)*sx(2));

% Plot the results.

figure(1), orient tall

% Plot the predistorter model.

subplot(211)
plot(distdata,'o'), hold on,plot(xi,':'),plot(rot90(xi),':'),plot(dd,'-'),
plot(rot90(dd),'-'),aspect1
title(['Nonlinearity order = ',num2str(ord-1),', Data points = ',num2str(points)])

% Plot the emitted signal constellation output by the high power amplifier.

subplot(212)
plot(yyv,'o'), hold on,plot(xi,':'),plot(rot90(xi),':'),plot(yyv,'-')
plot(rot90(yyv),'-'),aspect1
title(['F7.m, HPA Output, MSE = ',num2str(msefinal)])

```

## 2. FUNCTION FIX7.M.

```

function [distdata,yyv,g1,g3,g5,g7,i1,i3,i5,i7] = fix7(ord,x,points)

% function [distdata,yyv,g1,g3,g5,g7,i1,i3,i5,i7] = fix7(ord,x,points)
%
% This function calculates the inverse of the nonlinear distortion caused by high power
% amplifiers using a fixed Volterra Series approximation. The inverse model calculated
% is used to create a predistorter to correct the amplifier's distortion. This function is
% called by the script file f7.m. It in turn calls the function order7.m to determine the
% nonlinear data terms used to build the data matrix used in the solution of the Volterra
% kernel coefficients. The Volterra kernel coefficients are solved using a Least Squares
% technique.
%
% Input: The memory of the system (ord), the signal constellation in vector form (x), and
% the number of data points to be used in the Least Squares solution..
%
% Output: The predistorter model in vector form (distdata), the Volterra kernel coefficients

```

```

%          (g1, g3, g5, and g7), and the counters for the number of nonlinearity terms
%          (i1, i3, i5, and i7).
%

%   Generate the random vector of data points to be used in the Least Squares solution.

ra=rand(1,points);d=zeros(1,points);

%   Establish a table and assign the random number vector's elements to a point in the
%   signal constellation.

TAB = [linspace(0,1,64)' (1:64)'];
zz = fix(table1(TAB,ra));
d = x(zz);

%   Initialize a shift register with the number of states equal to ord.

sr = d(1:ord)

%   Simulate the distortion using Saleh's equations of the data and the signal constellation.

ysaleh = hpa(conj(x'));
ysalehd = hpa(conj(d'));

%   Create counters for the length of the data and the signal constellation.

[m n] = size(ysalehd)
[nx mx] = size(x);

%   Build the nonlinear data matrix, X, by shifting in the data. One column for each data point.

for k = 1:m

    sr = [ysalehd(k) sr(1,1:ord-1)];

%   Call the function order7.m to build a row of the data matrix.

    [y,i1,i3,i5,i7] = order7(sr,ord);
    xmat(k,:) = conj(y');
end

%   Determine the pseudoinverse of the nonlinear data matrix.

Xinv = inv(xmat'*xmat)*xmat';

%   Solve for the G Kernel coefficients. These are the Volterra Series coefficients to

```

% approximate the inverse model of the high power amplifier's distortion characteristics.

$g = X_{\text{inv}} * \text{conj}(d')$ ;

% Create an index of the size of the Volterra kernel coefficient matrix.

$[n1 \ m1] = \text{size}(g)$ ;

% Organize into the  $g$  coefficient vectors. One vector for each order of nonlinearity.

$g1 = (g(1:\text{ord}))$ ;

$g3 = [g(i1+1:i3)]$ ;

$g5 = [g(i3+1:i5)]$ ;

$g7 = [g(i5+1:i7)]$ ;

% Simulate the predistortion using the Volterra Series.

% Reinitialize the shift register to input the QAM signal constellation.

$r = \text{zeros}(1, \text{ord})$ ;

$yv = \text{zeros}(1, m)$ ;

% Iteratively shift in each point of the signal constellation.

for  $k = 1:64$

$r = [x(k) \ r(1, 1:\text{ord}-1)]$ ;

% Create the nonlinearity data matrix vector, called  $r_{\text{working}}$  in this call.

$[ \ r_{\text{working}}, i1, i3, i5, i7] = \text{order7}(r, \text{ord})$ ;

% Establish counters for each nonlinearity order.

$y1 = 0; y3 = 0; y5 = 0; y7 = 0$ ;

% Calculate the first order terms.

$r1 = r_{\text{working}}(1:\text{ord})$ ;

$y1 = r1' * g1$ ;

% Calculate the third order terms.

$r3 = r_{\text{working}}(i1+1:i3)$ ;

$y3 = r3' * g3$ ;

% Calculate the fifth order terms.

```

r5 = rworking(i3+1:i5);
y5 = r5' * g5;

% Calculate the seventh order terms.

r7 = rworking(i5+1:i7);
y7 = r7' * g7;

% Calculate the distorter data term for the signal constellation using the first through seventh
% order nonlinearity terms. This is for only one signal constellation point. One iteration
% must be performed for each point in the signal constellation.

distdata(1,k) = y1 + y3 + y5 + y7;

end

```



## APPENDIX D. MATLAB COMPUTER CODE FOR DETERMINING THE PERFORMANCE OF A PREDISTORTER MODEL.

### 1. SCRIPT F7M1.M.

```
% Script f7m1.m
%
% This script file simulates a 64 QAM communications system to determine the effectiveness
% of a predistorter model. The predistorter is employed as a table look up function and is
% calculated separately.
%
% Input: Predistorter model loaded from a .mat file.
%
% Output: Performance curve, Probability of symbol error versus Signal to Noise Ratio (SNR),
%         in vector and graphical form.
%

% Load the predistorter.
load f7m3

% Reset the seed of the random number generator to employ Monte Carlo principles in
% the system performance.

rand('seed',sum(100*clock));

% Generate vector of the SNR values to be evaluated.

vec = 2.5:2.5:35;

% Initialize the system parameters. Power of a symbol.

power = .4286*50;
Pb_vec = zeros(1,length(vec)); % Probability of symbol error.
L = 6; % Number of bits per symbol.
a = 1; % Amplitude of symbol power.
fc = 455000;fs = 4550000; % Carrier frequency, Sampling frequency.
k = 100000; % Number of symbols to transmit.
w = 2*pi*fc/fs; % Omega.
noperiods = 10; % Number of periods a symbol is transmitted.
nosamples = 10; % Number of samples per period a symbol is transmitted.
n = 0:noperiods*nosamples-1; % Vector for sampling times.
distdata1 = reshape(distdata,8,8); % Transform predistorter vector to an 8 x 8 matrix.
distdata1 = flipud(distdata1); % Rearrange to proper form.
b1 = fir1(5,.2); % FIR filter coefficients to remove 2fc components.

% Iteratively determine performance for each SNR level. Each level transmits k symbols
```



```

% and determines symbol error rate.

for num=1:length(vec)

% Set error to zero for each SNR.
error = 0;
yyv = conj(yyv);

% Transmit the symbols one at a time.

for l = 1:k

% Generate the bit stream for a symbol.

d = rand(1,L);d = d<.5;

% Shift in L bits and separate into the in-phase and quadrature components.

inphase = d(1:3);quad = d(4:6);

% Determine the codeword of the symbol. The values are real.

xin = table(inphase);
xq = table(quad);

% Build the QAM symbol.

sym = xin + i*xq;

% Predistort the symbol using the predistorter loaded earlier in a table look up form.

TABh = [(-.707:.202:.707)' (1:8)'];
TABv = [(.707:-.202:-.707)' (1:8)'];
hor = floor(table1(TABh,real(sym)));
vert = floor(table1(TABv,imag(sym)));

% Build the final predistorted symbol to be transmitted.

distsym = distdata1(vert,hor);

% Send the symbol through the HPA to simulate the distortion.

output=hpa(distsym);

% Pulse shape the symbol. Treat each branch independently.

```

```

xipre = real(output).*ones(1,noperiods*nosamples);
xqpre = imag(output).*ones(1,noperiods*nosamples);
% Modulate the signal.

xipreg= xipre.*cos(w.*n);
xqpreg= xqpre.*sin(w.*n);

% Add AWGN.
SNR_dB = vec(num);
snr = 10^(SNR_dB/10);
noise_var = power/(2*snr);
xipreg = xipreg + sqrt(noise_var/2).*randn(1,length(xipreg));
xqpreg = xqpreg + sqrt(noise_var/2).*randn(1,length(xipreg));

% Sum the two branches of the QAM transmitter and generate the transmitted signal.

output1 = xipreg-xqpreg;

% Signal is received and now demodulated.

rec = output1;

% Mix the received signal through both branches to remove the orthogonal signal.

yc = rec.*cos(w.*n);
ys = rec.*sin(w.*n);

% Filter the 2fc component.

ycfilt = filter(b1,1,yc);
ysfilt = filter(b1,1,ys);

% Send the signal through the summer.

ycth = (2/length(ycfilt))*sum(ycfilt);
ysth = (2/length(ysfilt))*sum(ysfilt);

% Build the received signal into the received symbol for detection.

y = ycth - ysth*i; %Negative to account for the change in threshold level.

% Estimate the symbol transmitted using a "choose the largest strategy".

sest = mqam_det(y,L,a);

```

```

% Determine if an error has been made. If so, track the error.

    if sest ~= sym,
        error = error + 1;
    end
end

% Determine the final probability of symbol error for the SNR value used.

Pb_vec(num) = error/k;

end

% Save the probability of symbol error and SNR vectors.

save f7m1d Pb_vec vec

```

## 2. FUNCTION TABLE.M.

```

function y = table(x)

% function y = table(x)
%
% This function serves as a lookup table used by both the
% in phase and quadrature phase branches to logically determine
% the amplitudes to associate with a given bit stream.
%
% Input: A real vector with 3 bit values.
%
% Output: Amplitude value assigned to the bit stream by the gray code.
%

[m,n]=size(x);
for k=1:m
    if x(k,:) == [0 0 0], y(k,1) = 0.101;

    elseif x(k,:) == [0 0 1], y(k,1) = -0.101;

    elseif x(k,:) == [0 1 0], y(k,1) = -0.505;

    elseif x(k,:) == [0 1 1], y(k,1) = -0.303;

    elseif x(k,:) == [1 0 0], y(k,1) = 0.303;

    elseif x(k,:) == [1 0 1], y(k,1) = 0.505;

```

```

elseif x(k,:) == [1 1 0], y(k,1) = -0.707;

elseif x(k,:) == [1 1 1], y(k,1) = 0.707;

end

end

```

### 3. FUNCTION MQAM\_DET.M

This function was written by Roy Axford, of NCCOSC, and provided by LCDR Bruce Watkins, of NCCOSC, San Diego, CA.

```

function [y] = mqam_det(x,L,A)
%*****
% MQAM_DET - Maximum-Likelihood M-ary QAM Symbol Detector.
%      (M = 2^L where L = no. of bits/symbol)
%
% INPUT:
% x      - Nx1 vector of (noisy) M-QAM symbols (complex numbers)
% L      - bits/symbol (= log2(M) where M is the M in M-ary)
%      There are M = 2^L symbols (points) in the constellation.
% A      - amplitude parameter (real valued scalar)
%
% Note: 'L' and 'A' define the constellation that was used by the
% transmitter as embodied in 'mqam_src.m' or 'mqam_enc.m'.
% However, here it will often be that A = 1. The equalizer
% preceding 'mqam_det.m' will adjust its gain appropriately
% during the training period. If the equalizer is blind, it
% will also adjust its gain such that the dispersion (a second
% and fourth moment related quantity) is consistent with the
% constellation being transmitted, if known, or ... we'll see.
% RAA 8:29AM 12/17/92
%
%
% OUTPUT:
% y      - Nx1 vector of M-QAM symbol decisions (complex numbers)
%
% ALGORITHM:
%
% Constructs constellation points (symbols) in the same way that
% 'mqam_src.m' and 'mqam_enc.m' do.
%
% Detection Criterion: Maximum-Likelihood Slicer. [1]
%      Each element of 'x' is mapped to the closest
%      (Euclidean distance) element of 'symbol'.

```

```

%
%           |           |
% Noisy QAM Symbol(s) ---> | mqam_det.m | ---> QAM Symbol Decision(s)
%           |           |
%
%
% M-FUNCTIONS CALLED BY MQAM_DET.M:
% none
%
% REFERENCES:
% [1] C.M. Thomas, M.Y. Weidner, and S.H. Durrani, "Digital Amplitude-
%   Phase Keying with M-ary Alphabets", IEEE Trans. Commun., vol. COM-22,
%   no. 2, pp. 168-180, Feb. 1974.
%
% [2] R.D. Gitlin, J.F. Hayes and S.B. Weinstein, "Data Communications
%   Principles", Plenum Press, 1992.
%
% [3] J.G. Proakis, "Digital Communications, 2nd Ed.", McGraw-Hill, 1989.
%
% [4] G.D. Forney et al., "Efficient Modulation for Band-Limited Channels,"
%   IEEE Journ. on Selected Areas in Commun., vol. SAC-2, no. 5,
%   pp. 632-647, Sep. 1984.
%
% [5] M. Stojanovic, J. Catipovic, and J.G. Proakis, "An Algorithm for
%   Multichannel Coherent Digital Communications Over Long Range
%   Underwater Acoustic Telemetry Channels," Proc. IEEE Oceans-'92.
%
% [6] R.L. Cupo and R.D. Gitlin, "Adaptive Carrier Recovery Systems for
%   Digital Data Communications Receivers," IEEE Journ. on Selected
%   Areas in Communications, vol. 7, no. 9, pp. 1328-1339, Dec. 1989.
%
% Written by: Roy A. Axford (axfordra@manta.nosc.mil)
% Last Update: 4:50PM 12/17/92
%
% Usage: y = mqam_det(x,L,A)
% where: x=input vector, L=log2(M) bits/symbol, A=amplitude factor
% (L=3 -> M=8) (L=4 -> M=16) (L=5 -> M=32)
% (L=6 -> M=64) (L=7 -> M=128) (L=8 -> M=256)
%*****

%*****
% input syntax checking
if (nargin ~= 3)
    error('Wrong number of inputs to m-function mqam_det. It wants 3.')
end
if (L ~= 3) & (L ~= 4) & (L ~= 5) & (L ~= 6) & (L ~= 7) & (L ~= 8),

```

```

    error('L restricted to 3, 4, 5, 6, 7 or 8 in m-function mqam_det.')
end
if (nargout ~= 1)
    error('Wrong number of outputs asked of m-function mqam_det. It gives 1.')
end
%
%*****
% initialize some vectors
j = sqrt(-1);
N = length(x);    % number of input noisy symbols (OK if N == 1)
y = zeros(N,1);   % column vector for the QAM symbol decisions
x_bits = zeros(1,L); % row vector "window" on x for bit groups
%
%*****
% "L-specific" sections of code that build the appropriate constellation.
%*****
%
% +++ Begin 8-QAM part
+++++
%
if L == 3, % 8-QAM (See fig. 3 of [4], or fig. 5.33 on p. 342 of [2].
    % This particular constellation was used in an
    % experimental underwater acoustic digital comm study
    % reported in [5].)
    % Note : The 999's are just place holders. It is highly
    % unlikely that a noisy symbol would be closer to
    % A*999 than to one of the actual constellation points.
    %
    symbol = A.*[ 999    999    j*3    999    999;
                  999   -1+j*1    999    1+j*1    999;
                  -3     999    999    999     3;
                  999   -1-j*1    999    1-j*1    999;
                  999    999   -j*3    999    999];

    %
    %
% +++ End 8-QAM part
+++++
%
% +++ Begin 16-QAM part
+++++
%
elseif L == 4, % 16-QAM
    %
    symbol = A.*[ -3+j*3  -1+j*3  1+j*3  3+j*3;
                  -3+j*1  -1+j*1  1+j*1  3+j*1;
                  -3-j*1  -1-j*1  1-j*1  3-j*1;

```

```

        -3-j*3 -1-j*3 1-j*3 3-j*3];
%
%
% +++ End 16-QAM part
++++
%
% +++ Begin 32-QAM part
++++%
elseif L == 5, % 32-QAM "Cross" as in fig. 5.4.9 on p.498 of [3].
    %
    symbol = A.*[ 999 -3+j*5 -1+j*5 1+j*5 3+j*5 999;
        -5+j*3 -3+j*3 -1+j*3 1+j*3 3+j*3 5+j*3;
        -5+j*1 -3+j*1 -1+j*1 1+j*1 3+j*1 5+j*1;
        -5-j*1 -3-j*1 -1-j*1 1-j*1 3-j*1 5-j*1;
        -5-j*3 -3-j*3 -1-j*3 1-j*3 3-j*3 5-j*3;
        999 -3-j*5 -1-j*5 1-j*5 3-j*5 999 ];
%
% NOTE: The (1,1), (1,6), (6,1) and (6,6) elements of 'symbol'
% are not used. (the 999's in the corners
%
% +++ End 32-QAM part
++++
%
% +++ Begin 64-QAM part
++++
%
elseif L == 6, % 64-QAM.
    %
    symbol = zeros(8,8);
    i = 0;
    for Rea = -.707:.202:.707
        i = i + 1;
        q = 0;
        for Ima = .707:-.202:-.707
            q = q + 1;
            symbol(q,i) = Rea + j*Ima; % This might look funny because the
                % quadrature coordinate 'q' corresponds
                % to a row which is the first index.
        end
    end
    symbol = A.*symbol;
%
%
% +++ End 64-QAM part
++++
%

```

```

% +++ Begin 128-QAM part
+++++
%
elseif L == 7, % 128-QAM. (See fig. 5.4.9 on p.498 of [3].)
%
% Also, a perhaps more practical 128-QAM constellation is shown on
% p. 1337 of [6] with varying degrees of distortion caused by
% phase jitter. RAA
%
% Create the 128-QAM constellation symbols.
% Some of the symbols won't be used (i.e. the "corners" of the cross).
% These square shaped regions are (q,i) = (1:2, 1:2),
% (11:12, 1:2),
% (1:2, 11:12),
% (11:12, 11:12).
%
symbol = zeros(12,12);
i = 0;
for Rea = -11:2:11
    i = i + 1;
    q = 0;
    for Ima = 11:-2:-11
        q = q + 1;
        symbol(q,i) = Rea + j*Ima; % This might look funny because the
            % quadrature coordinate 'q' corresponds
            % to a row which is the first index.

        % If we're in one of those corner areas, we need place holders:
        if ((q==1)&(i==1))|((q==1)&(i==2))|((q==2)&(i==1))|((q==2)&(i==2))
            symbol(q,i) = 999;
        elseif ((q==11)&(i==1))|((q==11)&(i==2))|((q==12)&(i==1))|((q==12)&(i==2))
            symbol(q,i) = 999;
        elseif ((q==1)&(i==11))|((q==1)&(i==12))|((q==2)&(i==11))|((q==2)&(i==12))
            symbol(q,i) = 999;
        elseif ((q==11)&(i==11))|((q==11)&(i==12))|((q==12)&(i==11))|((q==12)&(i==12))
            symbol(q,i) = 999;
        end
    end
end
symbol = A.*symbol;
%
% +++ End 128-QAM part
+++++
%
% +++ Begin 256-QAM part
+++++
%

```



```

elseif L == 8, % 256-QAM.
    %
    symbol = zeros(16,16);
    i = 0;
    for Rea = -15:2:15
        i = i + 1;
        q = 0;
        for Ima = 15:-2:-15
            q = q + 1;
            symbol(q,i) = Rea + j*Ima; % This might look funny because the
                                     % quadrature coordinate 'q' corresponds
                                     % to a row which is the first index.
        end
    end
    %
    symbol = A.*symbol;

% +++ End 256-QAM part
+++++
%
end % This 'end' is for the if's and elseif's above regarding the value
    % of L, the number of bits per symbol.
%
% So, at this point we've constructed the appropriate constellation.
% Now, for each x(n) we calculate the Euclidean distance between x(n) and
% each symbol in the constellation. We decide for the symbol that is
% closest to x(n) and assign this symbol to y(n).
% (Ref.: MATLAB Manual, Reference Section "max, min".)
%
for n = 1:N
    [row_of_column_mins what_rows] = min(abs(x(n) - symbol));
    [min_dist i_coord_of_closest] = min(row_of_column_mins);
    q_coord_of_closest = what_rows(i_coord_of_closest);
    %
    % quadrature coord. is row index
    % in-phase coord. is column index
    y(n) = symbol(q_coord_of_closest, i_coord_of_closest);
end
%
% /* ----- end of m-function mqam_det.m ----- */

```

## APPENDIX E. MATLAB CODE FOR DETERMINING THE ADAPTIVE PREDISTORTER AND COMMUNICATIONS SYSTEM PERFORMANCE.

### 1. SCRIPT FILE A7.M.

```
% Script file a7.m
%
% This script file calculates the inverse model of the high power amplifier's distortion
% characteristics using the Recursive Least Squares algorithm. The coefficients are
% based upon the Volterra kernel coefficients seen in other code used in this thesis.
% This script file's functions use the same function order7.m to determine the nonlinear
% data coefficients used in the RLS algorithm. The functions called by this script include
% ad7.m, berold.m, and ad7mod.m. These functions are used to determine the initial pre-
% distorter model, determine the communications system performance, and update the
% predistorter at a specified interval.
%
% Input: The signal constellations for QAM are loaded from a .mat file.
%
% Output: The predistorter model and communication system performance.
%

% Reset the seed of the random number generator to employ Monte Carlo theory.

rand('seed',sum(100*clock));

% Identify the memory of the system, ord - 1.

ord = 2;

% Load the QAM signal constellations.

load qam

% Identify the signal constellation to be employed and transformed into vector form.

xi = c64;
x=reshape(xi,1,64);

% Identify the number of iterations to be used in generating the predistorter.

points = 1000;

% Generate a random vector to assign symbols from the signal constellation.

ra=rand(1,points);
```

```

% Calculate first version of the predistorter.

[distdata,P,g1,g3,g5,g7,i1,i3,i5,i7] = ad7(ord,x,ra,points);

% Identify the range of signal to noise ratios (SNR) to be used to determine the
% communications system performance.

vec = 2.5:2.5:35;

% Initialize the probability of symbol error vector.

Pb_vec = zeros(1,length(vec));

% Call BEROLD.M. This will transmit data for a specified number of bits through the
% communications system. Then the predistorter is updated and transmission resumed.

% Identify the number of times the predistorter is to be updated.

update = 20;

% Initialize a matrix to track the error performance of each update to the predistorter and a
% vector to track the mse performance.

Pbsub1=zeros(update,length(vec));
msetrack=zeros(1,update);

% Begin the iterative transmission of symbols and periodic updating of the predistorter.

for up = 1:update

% Reshape the predistorter model into a vector and distort with the hpa model.

dd = reshape(distdata,8,8);
yyv=hpa(dd);

% Determine the mean square error of the predistorter estimate.

se = sum(sum(abs(yyv-conj(xi)).^2));sx=size(xi);
mse=se/(sx(1)*sx(2));
msetrack(1,up)=mse;

% Simulate the transmission of data by calling BEROLD.M.

[buffer,Pbsub] = berold(dd,vec);

% Update the Pb for the system.

```

```

Pbsub1(up,:)=Pbsub
Pb_vec = Pb_vec + Pbsub;

% Update the predistorter constellation.

[distdata,P,g1,g3,g5,g7,i1,i3,i5,i7] = ad7mod(ord,x,buffer,points,P);
end

% Determine the final probability of symbol error.

Pb_vec=Pb_vec/update;

% Reshape the predistorter for plotting.

distdata=conj(distdata);dd=reshape(distdata,8,8);

% Plot the Predistorter, HPA output and performance of the system.

figure(1), orient tall,
subplot(211)
plot(distdata,'o'), hold on,plot(dd),plot(rot90(dd))
axis equal
title('Predistorter Output')

% Send predistorted signal through the Saleh equations and arrange in an 8x8 matrix..

yv = hpa(distdata);
yyv=reshape(yv,8,8);

% Plot the HPA output.

subplot(212)
plot(yv,'o'), hold on,plot(yyv),plot(rot90(yyv))
axis equal
title(['HPA Output, delta = 15,lambda= .995, points = ',num2str(points)])

% Performance of the communications system.

figure(2),orient tall
semilogy(vec,Pb_vec),title('Adaptive Predistorter (7th Order) Comms System Performance')
xlabel('SNR'),ylabel('Pb')
etime(clock,t)

```

## 2. FUNCTION AD7.M.

```
function [distdata,P,g1,g3,g5,g7,i1,i3,i5,i7] = ad7(ord,x,ra,points)

% function [distdata,P,g1,g3,g5,g7,i1,i3,i5,i7]=ad7(ord,x,ra,points)
%
% This function calculates the inverse of the nonlinear distortion caused by the high power
% amplifier using the Recursive Least Squares (RLS) algorithm. The inverse model
% calculated is used to create a predistorter to correct the amplifier's distortion. This
% function is called by the script file a7.m. It in turn calls the function order7.m to determine
% the nonlinear data coefficients to build the column vector used as the models input. The
% Volterra kernel coefficients are solved recursively by the RLS algorithm.
%
% Input: The memory of the system (ord), the signal constellation in vector form (x),
% the vector of random numbers (ra), the number of recursive steps to obtain the
% predistorter constellation (points).
%
% Output: The predistorter in vector form (distdata), the inverse correlation matrix (P),
% the Volterra kernel coefficients (g1,g3,g5,g7), and the counters for the number of
% nonlinear terms (i1,i3,i5,i7).
%
% Establish a table and assign the random vector's elements to a point in the signal
% constellation.

TAB = [linspace(0,1,64)' (1:64)'];
zz = fix(table1(TAB,ra));
d = x(zz);

% Initialize a shift register with the number of states equal to ord..

sr = zeros(1,ord);

% Simulate the distortion, using Saleh's equations, of the data and signal constellation.

ysaleh = hpa(conj(x'));
ysalehd = hpa(conj(d'));

% Create indices of the length of the data and signal constellation.

[m n] = size(ysalehd)
[nx mx] = size(x);

% Perform one iteration of y to determine length of the input data vector and initialize P.

[y,i1,i3,i5,i7] = order7(sr,ord);
[leny,widy] = size(y);
```

```

% Initialize the RLS Parameters.

P = eye(leny,leny);
lambda = .995;
delta = 15;
g = zeros(leny,1);
P = (1/delta)*P;
e = zeros(m,1);
z=0;% count for mse

% Build the input vector, y, by shifting in the data.

for l = 1:m

    sr = [ysalehd(l) sr(1,1:ord-1)];

    [y,i1,i3,i5,i7] = order7(sr,ord);

% Execute the RLS algorithm.

    K = ((1/lambda)*P*y)/(1+(1/lambda)*y'*P*y);
    e(l) = d(l) - g'*y;
    g = g + K*conj(e(l));
    P = (1/lambda)*P-(1/lambda)*K*y'*P;

end

% Organize into the g coefficient vectors. One vector for each order of nonlinearity.

g1 = (g(1:ord));
g3 = [g(i1+1:i3)];
g5 = [g(i3+1:i5)];
g7 = [g(i5+1:i7)];

% Simulate the distortion using the Volterra Series.
% Reinitialize the shift register to input the QAM signal constellation.

r = zeros(1,ord);

% Iteratively shift in each point of the signal constellation.

for k = 1:64

    r = [x(k) r(1,1:ord-1)];

```

```

% Create the nonlinearity data vector, called rworking in this call.

[rworking,i1,i3,i5,i7] = order7(r,ord);

% Initialize values for each nonlinearity order.

y1 = 0;y3 = 0;y5 = 0;y7 = 0;

% Calculate the first order terms.

r1 = rworking(1:ord);
y1 = r1'*g1;

% Calculate the third order terms.

r3 = rworking(i1+1:i3);
y3 = r3' * g3;

% Calculate the fifth order terms.

r5 = rworking(i3+1:i5);
y5 = r5' * g5;

% Calculate the seventh order terms.

r7 = rworking(i5+1:i7);
y7 = r7' * g7;

% Calculate the predistorter data term for the signal constellation using the first through
% seventh order nonlinearity terms. This is for only one signal constellation point. One
% iteration must be performed for each point in the signal constellation.

distdata(1,k) = y1 + y3 + y5 + y7;

end

```

### 3. FUNCTION BEROLD.M.

```
function [buffer,Pbsub] = berold(dd,vec)
```

```

% function [buffer,Pbsub] = berold (dd,vec)
%
% This function simulates the 64 QAM communications system to determine the effectiveness
% of the predistorter for each iteration of the update. The predistorter is employed as a
% table look up function and is calculated with the functions ad7.m and ad7mod.m.
%

```

```

% Input: Predistorter model (dd), a dn Signal-to-Noise ratio range (vec).
%
% Output Last data symbols transmitted to be used in the RLS update, the length of the vector
% is specified internally. (buffer) The probability symbol error for the current
% iteration of berold.m (Pbsub).

% Initialize the system.

power = .4286*100/2;           % Power of the symbol.
Pbsub = zeros(1,length(vec)); % Probability of symbol error vector.
L = 6;                         % number of bits/symbol.
a = 1;                         % amplitude of the signal.
fc = 455000; fs = 4550000;     % carrier frequency, sampling frequency.
k=1000;                        % number of symbols to transmit.
w = 2*pi*fc/fs;                % Omega.
noperiods = 10;                % number of periods per symbol.
nosamples = 10;                % number of samples per period.
n = 0:noperiods*nosamples-1;   % vector for samplig times.
distdata1 = conj(dd)           % Rearrange for proper form.
buffer = zeros(1,1000);        % Output vector with last symbols transmitted.
b1 = fir1(5,.2);               % FIR filter coefficients to remove 2fc components.

% Iteratively determine the performance for each SNR level. Each level transmits k symbols
% determines the symbol error rate.

for num=1:length(vec)

% Set the error to zero.

    error = 0;
    yyv = conj(yyv);

% Transmit the symbols one at a time.

    for l = 1:k

% Generate the bit stream for the symbol.

        d = rand(1,L); d = d<.5;

% Shift in L bits and separate into the in-phase and quadrature components.

        inphase = d(1:3); quad = d(4:6);

% Determine the codeword of the symbol. The values are real..

```



```

    xin = table(inphase);
    xq = table(quad);

% Build the QAM symbol.

    sym = xin + i*xq;

% Predistort the symbol using the predistorter loaded earlier in a table look up form.

    TABh = [(-.707:.202:.707)' (1:8)'];
    TABv = [(.707:-.202:-.707)' (1:8)'];
    hor = floor(table1(TABh,real(sym)));
    vert = floor(table1(TABv,imag(sym)));

% Build the final predistorted symbol to be transmitted.

    distsym = distdata1(vert,hor);

% Send the symbol through the HPA to simulate the distortion.

    output=hpa(distsym);

% Pulse shape the symbol.

    xipre = real(output).*ones(1,noperiods*nosamples);
    xqpre = imag(output).*ones(1,noperiods*nosamples);

% Modulate the symbol.

    xipreg= xipre.*cos(w.*n);
    xqpreg= xqpre.*sin(w.*n);

% Add AWGN.
    SNR_dB = vec(num);
    snr = 10^(SNR_dB/10);
    noise_var = power/(2*snr);
    xipreg = xipreg + sqrt(noise_var/2).*randn(1,length(xipreg));
    xqpreg = xqpreg + sqrt(noise_var/2).*randn(1,length(xipreg));

% Sum the two branches of the QAM transmitter and generate the transmitted signal.

    output1 = xipreg+xqpreg;

% Demodulate the received signal.

    rec = output1;

```

```

% Mix the received signal through both branches to remove the orthogonal signal.

yc = rec.*cos(w.*n);
ys = rec.*sin(w.*n);

% Filter the 2fc component.

ycfilt = filter(b1,1,yc);
ysfilt = filter(b1,1,ys);

% Send the signal through the summers.

ycth = (2/length(ycfilt))*sum(ycfilt);
ysth = (2/length(ysfilt))*sum(ysfilt);

% Build the received signal into the received symbol for detection.

y = ycth + ysth*i;

% Estimate the symbol transmitted using a "choose largest strategy" and determine if in
% error.

sest = mqam_det(y,L,a);
if sest ~= sym,
    error = error + 1;
end

% Determine if the symbol must be buffered.

if l > k-1000,buffer(1,l-k+1000) = sym;end
end

% Determine the probability of error for this iteration.

Pbsub(num) = error/k;

end

```

#### 4. FUNCTION AD7MOD.M.

```
function [distdata,P,g1,g3,g5,g7,i1,i3,i5,i7] = ad7mod(ord,x,buffer,points,P)
```

```
% function [distdata,P,g1,g3,g5,g7,i1,i3,i5,i7] = ad7mod(ord,x,buffer,points,P)
```

```
%
```

```
% This function calculates the inverse of the nonlinear distortion caused by the high power
% amplifier using the Recursive Least Squares algorithm. This function is similar to the
```

```

% function ad7.m but has been modified. The modifications include passing the buffer and
% previous inverse correlation matrix, P. This function is called by the script file a7.m and
% calculates the inverse distortion model.
%
% Input: The memory of the system (ord), the signal constellation in vector form (x), the
%        buffer from the immediately preceding communications system simulation (buffer),
%        the number of recursive steps to obtain the predistorter constellation (points), and
%        the previous inverse correlation matrix (P).
%
% Output: The predistorter in vector form (distdata), the inverse correlation matrix (P), the
%         Volterra kernal coefficients (g1,g3,g5,g7), and the counters for the number of
%         nonlinearity terms (i1,i3,i5,i7).

% Initialize a shift register with the number of states equal to ord.

sr = zeros(1,ord);

% Simulate the distortion, using Saleh's equations, of the data and signal constellations.

ysaleh = hpa(conj(x'));
ysalehd = hpa(conj(buffer'));

% Create indices of the length of the data and signal constellation.

[m n] = size(ysalehd)
[nx mx] = size(x);

% Perform one iteration of y to determine length of the input data vector.

[y,i1,i3,i5,i7] = order7(sr,ord);
[leny,widy] = size(y);

% Initialize the RLS parameters.

lambda = .995;
delta = 15;
g = zeros(leny,1);
e = zeros(m,1);

% Build the input vector, y, by shifting in the data.

for l = 1:m

    sr = [ysalehd(l) sr(1,1:ord-1)];
    [y,i1,i3,i5,i7] = order7(sr,ord);

```

% Execute the RLS algorithm.

```
K = ((1/lambda)*P*y)/(1+(1/lambda)*y'*P*y);  
e(l) = buffer(l) - g'*y;  
g = g + K*conj(e(l));  
P = (1/lambda)*P-(1/lambda)*K*y'*P;  
end
```

% Organize into the g coefficient vectors. One vector for each order of nonlinearity.

```
g1 = (g(1:ord));  
g3 = [g(i1+1:i3)];  
g5 = [g(i3+1:i5)];  
g7 = [g(i5+1:i7)];
```

% Simulate the Distortion using the Volterra Series.

% Reinitialize the shift register to input the QAM signal constellation.

```
r = zeros(1,ord);
```

% Iteratively shift in each point of the signal constellation.

```
for k = 1:64
```

```
    r = [x(k) r(1,1:ord-1)];
```

% Create the nonlinearity data vector, called rworking in this call.

```
[rworking,i1,i3,i5,i7] = order7(r,ord);
```

% Initialize values for each nonlinearity value.

```
y1 = 0;y3 = 0;y5 = 0;y7 = 0;
```

% Calculate the first order terms.

```
r1 = rworking(1:ord);  
y1 = r1'*g1;
```

% Calculate the third order terms.

```
r3 = rworking(i1+1:i3);  
y3 = r3'*g3;
```

% Calculate the fifth order terms.

```

r5 = rworking(i3+1:i5);
y5 = r5' * g5;

% Calculate the seventh order terms.

r7 = rworking(i5+1:i7);
y7 = r7' * g7;

% Calculate the predistorter data term for the signal constellation using the first through
% seventh order nonlinearity terms. This is for only one signal constellation point. One
% iteration must be performed for each point in the signal constellation.

distdata(1,k) = y1 + y3 + y5 + y7;

end

```

## LIST OF REFERENCES

1. Morris, N. H., *Industrial Electronics for Technicians and Technician Engineers*, New York: McGraw-Hill, 1970.
2. Terman, Frederick Emmons, et. al., *Electronic and Radio Engineering*, New York: McGraw-Hill, 1955.
3. DeFrance, J. J., *Communications Electronics Circuits*, Corte Madera, CA: Rinehart Press, 1972.
4. Sedra, Adel and Smith, Kenneth, *Microelectronics Circuits*, Fort Worth, TX: Saunders College Publishing, 1991.
5. Gottlieb, Irving M., *Practical RF Power Design Techniques*, New York: TAB Books, 1993.
6. Krauss, Herbert L., et. al., *Solid State Radio Engineering*, New York: John Wiley & Sons, 1980.
7. A. A. M. Saleh, "Frequency-Independent and Frequency-Dependent Nonlinear Models of TWT Amplifiers," *IEEE Trans. on Communications*, Vol. COM-29, pp. 1715 - 1720, Nov. 1980.
8. E. Biglieri, et. al., "Analysis and Compensation of Nonlinearities in Digital Transmission Systems," *IEEE Journal on Selected Areas in Communications*, Vol. 6, pp. 42-51, Jan 1988.
9. S. Benedetto, et. al., "Modeling and Performance Evaluation of Nonlinear Satellite Links: A Volterra Series Approach," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AAES-15, pp. 494-506, July 1979.
10. Hoile, G., et. al., "Nonlinear MESFET Model for the Design of RF Power Amplifiers," *IEE Proceedings-G*, Vol. 139, No. 5, pp. 574-580, Oct. 1992.
11. Hayashi, H., et. al., "Quasilinear Amplification using Self Phase Distortion Compensation Technique," *IEEE Transactions on Microwave Theory and Technology*, Vol. MTT-43, No. 11, pp. 2551-2564, Nov. 1995.
12. Fihel, A. and Sari, H., "Performance of Reduced-Bandwidth 16 QAM with Decision-Feedback Equalization," *IEEE Trans. on Communications*, Vol. COM-35, pp. 715-723, July 1987.
13. Cavers, J., "Amplifier Linearization Using a Digital Predistorter with Fast Adaptation and Low Memory Requirements," *IEEE Trans. on Vehicular Technology*, Vol. 39, No. 4, pp. 374-882, Nov. 1990.

14. Watkins, B., et. al., "Neural Network based Adaptive Predistortion for the Linearization of Nonlinear RF Amplifiers," *Proceedings of MILCOM95*, Nov. 1995.
15. Watkins, B. et. al., "Model Based Neural Network Predistortion of Nonlinear Amplifiers," *Proc. of International Conference on Neural Networks*, Nov. 1996.
16. Stapleton, S., et. al., "Simulation and Analysis of an Adaptive Predistorter Utilizing a Complex Spectral Convolution," *IEEE Trans. on Vehicular Technology*, Vol. 41, No. 4, pp. 387-394, Nov. 1992
17. Sklar, Bernard, *Digital Communications: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1988.
18. Frerking, Marvin E., *Digital Signal Processing in Communications Systems*, New York: Van Nostrand Reinhold, 1994.
19. Proakis, John G., *Digital Communications*, New York: McGraw-Hill, 1995.
20. Lazzarin, G., et. al., "Nonlinearity Compensation in Digital Radio Systems," *IEEE Trans. on Communications*, Vol. 42, pp. 988-999, April 1994.
21. Schetzen, Martin, "Nonlinear System Modeling Based on the Wiener Theory," *Proceedings of the IEEE*, Vol. 69, No. 12, Dec 1981.
22. Schetzen, Martin, *The Volterra and Wiener Theories of Nonlinear Systems*, Malabar, FL: Krieger Publishing, 1989.
23. Benedetto, S., et. al., *Digital Transmission Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
24. Rugh, Wilson, *Nonlinear System Theory - The Volterra/Wiener Approach*, Baltimore: Johns Hopkins University Press, 1981.
25. Faulkner, M. and Mattsson, T., "Spectral Sensitivity of Power Amplifiers to Quadrature Modulator Misalignment," *IEEE Trans. on Vehicular Technology*, Vol. 41, pp. 516-525, Nov. 1992.
26. Serfaty, S. et. al., "Cancellation of Nonlinearities in Bandpass QAM Systems," *IEEE Trans. on Communications*, Vol. 38, No. 10, Oct. 1990.
27. Therrien, Charles W., *Discrete Random Signals and Statistical Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1992.
28. Haykin, Simon, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1991.

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center..... 2  
8725 John J. Kingman Rd., STE 0944  
Ft. Belvoir, VA 22060-6218
  
2. Dudley Knox Library..... 2  
Naval Postgraduate School  
411 Dyer Rd.  
Monterey, CA 93943-5101
  
3. Chairman, Code EC..... 1  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, CA 93943-5121
  
4. Prof. Murali Tummala, Code EC/Tu..... 4  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, CA 93943-5121
  
5. Prof. Charles Therrien, Code EC/Ti..... 1  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, CA 93943-5121
  
6. Richard North..... 1  
NCCOSC RDT&E DIV  
53560 Hull St.  
San Diego, CA 92152-5001
  
7. LT Bruce E. Watkins..... 1  
NCCOSC RDT&E DIV  
53560 Hull St.  
San Diego, CA 92152-5001
  
8. MAJ Michael T. Donovan..... 2  
125 Leidig Circle  
Monterey, CA 93940